Neural Network Based Modeling and Data Mining of Blast Furnace Operations

by

Bradley S. Banks

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

September 1, 1999

Author______________________________
Department of Electrical Engineering and Computer Science
September 1, 1999

Certified by______________________________
Prof. Amar Gupta
Thesis Supervisor

Accepted by______________________________
Prof. Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Acknowledgements

The author wishes to sincerely thank the following people:

Prof. Amar Gupta, for providing direction and guidance on this project, and in general for being a supportive and helpful advisor;

Dr. Debashish Bhattacharjee, for all for his efforts to collect data for my research, and to provide answers to my numerous uninformed questions about blast furnace operations;

All the members of the MIT/Sloan datamining research group, including Angela Ge, Ravindra Sarma, Vladislav Gabovich, Ashish Mishra, and Shastri Sandy, for countless hours of processing data, running models, writing code, and without whom this thesis quite simply would have been impossible to do.

And finally, my deep personal gratitude and appreciation goes to my parents, Randal Banks and Lora Cooley, to the rest of my family, and to all my friends, for all their love and support through good times and bad.

Neural Network Based Modeling and Data Mining of Blast Furnace Operations

by

Bradley S. Banks

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

September 1, 1999

Abstract

Industrial production processes have traditionally been predicted using models whose structure is determined from domain specific knowledge. Steel production is an example of such a process, where conventional modeling techniques are based on knowledge of the materials and reactions involved. These models have several shortcomings, including the impossibility of constructing precise models, the inability to adapt to changing conditions, and the high cost of development. Because of these limitations, there is growing interest in applying methods which take a data driven approach to modeling these processes. One such method, the Artificial Neural Network (ANN), has been successfully used in many areas. ANNs offer a data driven approach which has potential to produce a more accurate, flexible model in less time and at a lower cost. This model can then be used for simulation, prediction, and control. A case study application of using ANNs to model the production of pig iron, an intermediate step in producing steel, is presented. Two qualitatively different aspects of pig iron production, hot metal temperature and hot metal silicon content, were modeled. Early results indicated the importance of preprocessing the raw data in order to make network learning easier. Modeling efforts focused on the most well known neural network architecture, a feed forward network trained with the backpropagation learning algorithm. A neural network software package, NNRUN, was developed to automate the search for the best neural network for a given dataset. The combination of preprocessing and modeling with NNRUN produced models which yielded significantly more accurate predictions than traditional modeling methods.

Table of Contents

I. Introduction

The past decade has seen a growing interest in using artificial neural networks to capture information contained in extremely large datasets. These efforts, collectively known as data mining, have been successful in many application domains. Fields as diverse as astrophysics (Burl 94), management (Bansal 98), and medicine (Akay 92), along with many others, have found data mining techniques to be a powerful tool for finding useful information hidden within massive databases.

These methods are also being used in industrial applications. Modeling, predicting, and controlling complex industrial processes are natural applications of machine learning methods. One of the most popular of these methods involves the use of artificial neural networks, or ANNs. This popularity is a result of several properties of ANN, including the ability to capture arbitrary nonlinear relationships, to handle data with many inputs, to handle more than one output variable, and to be used for either classification or regression problems. Other characteristics that often make neural networks a preferred choice are because they require no prior knowledge, and because ANNs are computationally efficient (Rumelhart 86).

Application Domain: Steel Production

The production of steel is an example of a complex industrial process. Modeling, predicting, and controlling this process has proven to be a difficult task. Although many decades, even centuries, have been spent trying to model this process using chemical reactions, thermodynamics, and mass-transfer equations, these approaches have had only limited success (Biswas 1984). The fundamental obstacle to such approaches is that such models depend on the values of many parameters which are impossible to know accurately. The extreme conditions found inside the blast furnace, which operates at temperatures near 1500° Celsius (2700° Fahrenheit), prevent the collection of data which are necessary for these deterministic models. There are other sources of uncertainty which limit the accuracy of these models, including variations in the composition of raw materials, the effects of random fluctuations within the blast furnace, and the effects of interactions between input variables (Ullmann 1985).

Although these deterministic techniques can provide limited guidance in producing steel, most modern steel plants are moving towards a data driven approach to modeling and controlling the production process. Being 'data driven' means a model of the process is built by using the actual data from the production process. The structure of such a model is quite flexible, and the parameters of the model are determined from analyzing the data. The advantage of being data driven is that such an approach can capture the actual behavior of the process represented by the data. This approach differs from traditional techniques, which produce rigid models that are based on theoretical knowledge of the physical and chemical properties involved in the production process. The data driven approach is more flexible because it is allowed to explore the space of all possible relationships between inputs and outputs, and the resulting model can adapt over time to changing conditions. These are properties which the traditional models lack.

Artificial neural network methods have been used successfully on several problems in steel manufacturing. Hirata et.al. developed a neural network based system coupled with an expert system to control aspects of blast furnace operations (Hirata 1990). Another group applied neural network techniques for process control and to help detect and diagnose abnormal faults (Ungar

1990). These results indicate that artificial neural networks are well suited for modeling blast furnace operations.

Also relevant to the work contained in this paper is research involving the use of ANNs for time series prediction. The production data from the blast furnace operation is a time-series, and thus methods for analyzing time-series data are appropriate. Neural networks have been used to model and predict time series from many application areas. In *Predicting the future and understanding the past*, edited by A. Weigend, a group of studies are presented that predicted time series from several application domains. Successful applications included modeling of an industrial process, as well as predicting time-series data from the financial and biological domains (Weigend 1993).

This paper presents the results of attempts to model and predict two important variables in blast furnace operations. These variables, Hot Metal Temperature (HMT) and Silicon Content, are reliable indicators of the quality of the blast furnace's output. The research is based on two months of operational data collected from the 'G' Blast Furnace of the Tata Iron and Steel Company (TISCO) in Jamshedpur, India. The 'G' Blast Furnace is a modern blast furnace which has embedded electronics for monitoring and control. These electronics allow data to be automatically collected and stored for future analysis.

Along with the results of the modeling efforts, a neural network software package is also presented. NNRUN is a complete data processing and modeling system which automates many aspects of using ANNs. Its features include the ability to find automatically the optimal neural network architecture for a dataset, integrated use of the resulting network model for predicting new data, and an easy to use menu based user interface.

Overview of Remaining Chapters

Chapter 2, "**Production of Pig Iron in Steel Manufacturing**", provides background information on the application domain, production of pig iron. An overview of the entire steel production process is presented, followed by a more detailed treatment of the blast furnace production of pig iron. The target output variables and key input variables are introduced and their importance is discussed.

Chapter 3, "**Artificial Neural Networks**", presents material on both the theory and application of ANNs. Industrial applications are discussed in some detail. Alternatives to neural network methods are presented and compared. Uses of ANNs for modeling, prediction, and control are presented.

Chapter 4, "**Data and Data Preprocessing**", describes the data and all aspects of how it was manipulated before training of the ANN models. The importance of preprocessing data is emphasized as a crucial step in the success of the modeling process.

Chapter 5, "**NNRUN: A Neural Network Software Package**", describes a software package developed by the research team to perform neural network based modeling and data mining. The features of the package are discussed, along with a high-level view of the system architecture and brief descriptions of the major pieces of the package.

Chapter 6, "**Results**", presents the results of modeling Hot Metal Temperature and Hot Metal Silicon Content. The outcomes of many alternative ways of preprocessing the data, along with various network architectures, are presented and compared. Results of predicting at several different time intervals into the future are discussed.

Chapter 7, "**Conclusions**", presents the conclusions to be drawn from the results of this work. Insights about the pivotal role that preprocessing can play in determining model accuracy are discussed. Ideas for future avenues of research are proposed. The thesis concludes with a summary of the new contributions contained in this work.

II. Production of Pig Iron in Steel Manufacturing

Steel production is a complex process involving many steps and several inputs. This chapter begins with an overview of pig iron and steel production, and then presents a more in depth discussion of the particular variables and processes analyzed in this study.

Overview of Pig Iron and Steel Production

There are several distinct steps involved in steel production. The raw materials of the process are sinter (which contains the iron ore), and coke (a fuel made from coal). The sinter and coke are combined in a blast furnace to produce molten iron, which is also known as pig iron or hot metal. Pig iron is the input to further steel manufacturing, which combines the pig iron with various alloys in either a basic oxygen vessel or an electric arc furnace in order to produce steel. A graphic depiction of this process can be seen in Figure 1 on the following page.

The Blast Furnace

The blast furnace is considered the key component of any steel mill. It is the central unit of the integrated steel production process. Inside the blast furnace, the oxygen from the iron oxides is removed to yield nearly pure liquid iron. This liquid iron, or pig iron, is the raw material used in the steel plants. As with any product, the quality of this pig iron can vary. The most important determinants of the quality are (1) the amount and composition of any impurities, and (2) the temperature of the hot metal when it is tapped from the blast furnace. The quality of the pig iron produced is important in determining how costly it will be to produce steel from the pig iron, as well as constraining what final types of steel into which the pig iron can be made. Predicting the quality of this pig iron is the focus of the work presented in this thesis.

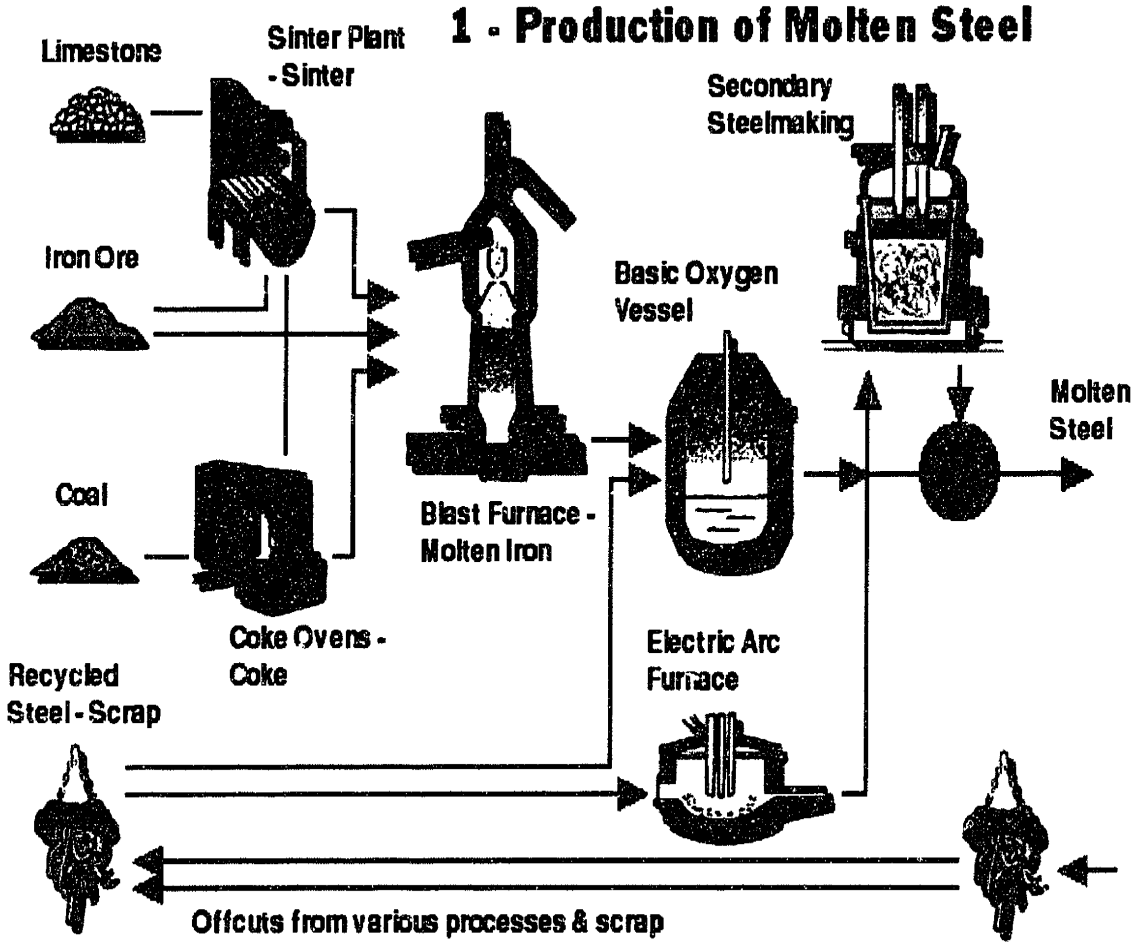


Figure 1: A Flow Diagram of Steel Production. Raw materials on the left are converted to coke and sinter, which are main inputs to the blast furnace. The output of the blast furnace is liquid iron (also known as pig iron or hot metal), which is then processed further to make steel.

The Process of Pig Iron Production

The production of pig iron begins with the making of the raw materials used in the process, coke and sinter. Coke plays two important roles in the blast furnace, first by providing the fuel for the process, and secondly as a reducing agent for the iron ore. Coke is produced by heating blended coal in coke ovens. Once carbonized, the coke is pushed out of the ovens and allowed to cool. Sinter contains the iron ore which will be converted to raw iron, known as pig iron. It is produced when fine-sized ore is mixed with coke and fluxes and heated in a sinter plant. Fluxes are added materials, principally limestone, which draw off impurities from the ore during the production of pig iron. The high temperatures generated fuse the ore particles and fluxes together to form porous nuggets called sinter. The use of sinter in the blast furnace helps make the ironmaking process more efficient.

Iron ore in lumps and pellets, coke, sinter and possibly extra flux are carried to the top of the blast furnace and then tipped, or charged, into the furnace. Hot air (900 degrees C etc) is blasted into the bottom of the furnace through nozzles called tuyeres. The oxygen in the air combusts with the coke to form carbon monoxide gas, and this generates a great deal of heat. Frequently oil or coal is injected with the air, which enables less (relatively expensive) coke to be used. The carbon monoxide flows up through the blast furnace and removes oxygen from the iron ores on their way down, thereby leaving iron. The heat in the furnace melts the iron, and the resulting liquid iron (or hot metal as it is called in the industry) is tapped at regular intervals by opening a hole in the bottom of the furnace and allowing it to flow out. The fluxes combine with the impurities in the coke and ore to form a molten slag, which floats on the iron and is also removed (tapped) at regular intervals.

The process described above goes on continuously for ten years or more. (This period is known as a campaign.) If the furnace were allowed to cool, damage would occur to its lining of refractory bricks as a result of their contracting as they cooled. Eventually the refractory brick linings are worn away, and at that stage the process is stopped and the furnace relined with new bricks, ready to begin its next campaign (Steel 99).

The rest of the steel manufacturing process involves further refining and processing of this pig iron. From the blast furnace, the pig iron is turned into steel either through a basic oxygen converter or an electric arc furnace. The BOS (Basic Oxygen Steelmaking) process is the major modern process for making bulk steels. The modern BOS vessel makes up to 350 tons of steel at a time, and the whole process takes about 40 minutes. Increasingly today, steels after they have been tapped (poured) from the furnace undergo a further stage of processing called secondary steelmaking before the steel is cast. This applies to both the basic oxygen process route and to the electric arc furnace route. The objective in all cases is to meet customer requirements for particular grades of steel by fine tuning the chemical composition of the steel and/or by improving homogenization of temperature (making sure that the steel is the same temperature throughout) and by removing impurities.

Pig Iron Quality: Impurities, Silicon Percentage, and Hot Metal Temperature

The pig iron produced in the blast furnace will always have some impurities. The level and composition of these impurities are important measures of the quality of the hot metal. High levels of impurities make it more expensive to convert the pig iron to steel, as much of the reducing process of the blast furnace must be repeated if impurity levels are too high.

The hot metal tapped from the blast furnace consists mainly of iron (about 94%). Impurities include about 4 to 5% carbon, 0.2 to 1% silicon, and possibly trace amounts of manganese, phosphorous, and titanium. These impurities tend to make the iron brittle. Steelmaking refines iron primarily by reducing its carbon content to make it a stronger and more useful product (Ullmann 85).

The most undesirable impurity in the pig iron is silicon. While the unwanted carbon can be removed in later stages of steel processing, silicon stays in solution in the molten iron and is not easily separated out. Removing excess silicon after the blast furnace stage is a time consuming and expensive process. Thus, modeling, predicting, and controlling the percentage of silicon is important in producing high quality pig iron during the blast furnace stage.

Silicon percentage deserves special attention not only because it is an important quality parameter, but also because it reflects the internal state of the high-temperature region of the furnace. Many of the dynamic changes in silicon content correspond to changes in the hot metal temperature and shifts in the heat level of the blast furnace. Silicon content provides useful information about these internal variables.

Difficulties in Predicting Silicon Percentage

Prediction of silicon percentage is a difficult problem for several reasons. Chemical models alone can not explain the process of silicon reduction quantitatively. The complex behavior of the process (due to unknown and dynamically changing lengths of residency, and the different conditions encountered as a charge passes from the top to the bottom of the blast furnace) as well as the lack of measurements of the internal conditions of the blast furnace, make mathematical modeling cumbersome, if not impossible. To further complicate matters, there are nonlinear relationships between input variables, silicon percentage, and hot metal temperature. These nonlinear relationships are difficult to include in static chemical models.

While traditional methods may be unable to model these relationships, there is hope. Recent advances in artificial intelligence and statistics have developed techniques that are well suited to handling problems such as the one described above. One of the foremost of these techniques, artificial neural networks, is the subject of the next chapter.

III. Artificial Neural Networks

The use of artificial neural networks has increased tremendously over the past 15 years. Neural networks have been used successfully to model problems in widely different fields, as discussed in the Introduction chapter of this thesis. This chapter provides relevant background information on neural network history and an introduction to the methods by which these networks learn.

Artificial and Biological Neural Networks

Much of the early development of artificial neural networks was motivated by the design of biological nervous systems, especially the brain. The brain consists of enormous numbers of very simple individual units, neurons, which are highly interconnected through synapses. These synapses transmit the effect of a neuron's firing to all of the neurons to which its output axon is connected. These synapses vary in the degree to which they effect the neurons into which they feed. Similarly, an artificial neural network consists of a number of simple processing units which are highly interconnected. The connection between a pair of nodes is directed and has a weight, which represents how much the output of one node influences the behavior of the node to which it serves as an input. In this analogy, the nodes of the artificial neural network correspond to the neurons of the brain, and the weights on the connections between nodes correspond to the strength of the synapses which connect neurons.

Because much of the early research into artificial neural networks attempted to mimic biological neural networks, there are a number of superficial similarities between the two, such as analogous structures and terminology. However, at a deeper, more fundamental level, artificial and biological neural networks have important differences and are nearly unrelated. As understanding of the mathematical properties of artificial neural networks has developed, they are increasingly seen as a member of a more general class of modeling methods known as statistical learning machines. Members of this group include such methods as linear regression, genetic algorithms (Goldberg 89), ID trees (Quinlan 93), support vector machines (Vapnik 95), as well as any number of other methods. In general, a statistical learning machine is considered to be any model which tries to predict one or more output variables from a possibly large number of input variables. These models contain parameters which are set by applying a learning algorithm to a dataset of example patterns. Such models share several important characteristics which makes them useful to consider together.

Feed Forward Neural Networks and Error Backpropagation

In this work, research focused on a well analyzed network architecture known as a feed-forward neural network (Rumelhart 1986). Although there are many possible network architectures, the feed-forward neural network model is by far the most commonly used. This type of network consists of layers of nodes where connections run only from earlier nodes to later nodes, and never from a later node to an early node. This guarantees the network will not have any feedback cycles. This property makes algorithms for training the networks, such as backpropagation, much easier to develop and analyze. This network architecture is pictured in Figure 2 on the following page.

The feed-forward network is trained by presenting the system with a dataset of training examples. Each example includes values of the input variables as well as the corresponding output. During training, an example pattern is presented to the network. Using the current set of weights, the network computes its predicted output for the given values of the input variables. The actual value

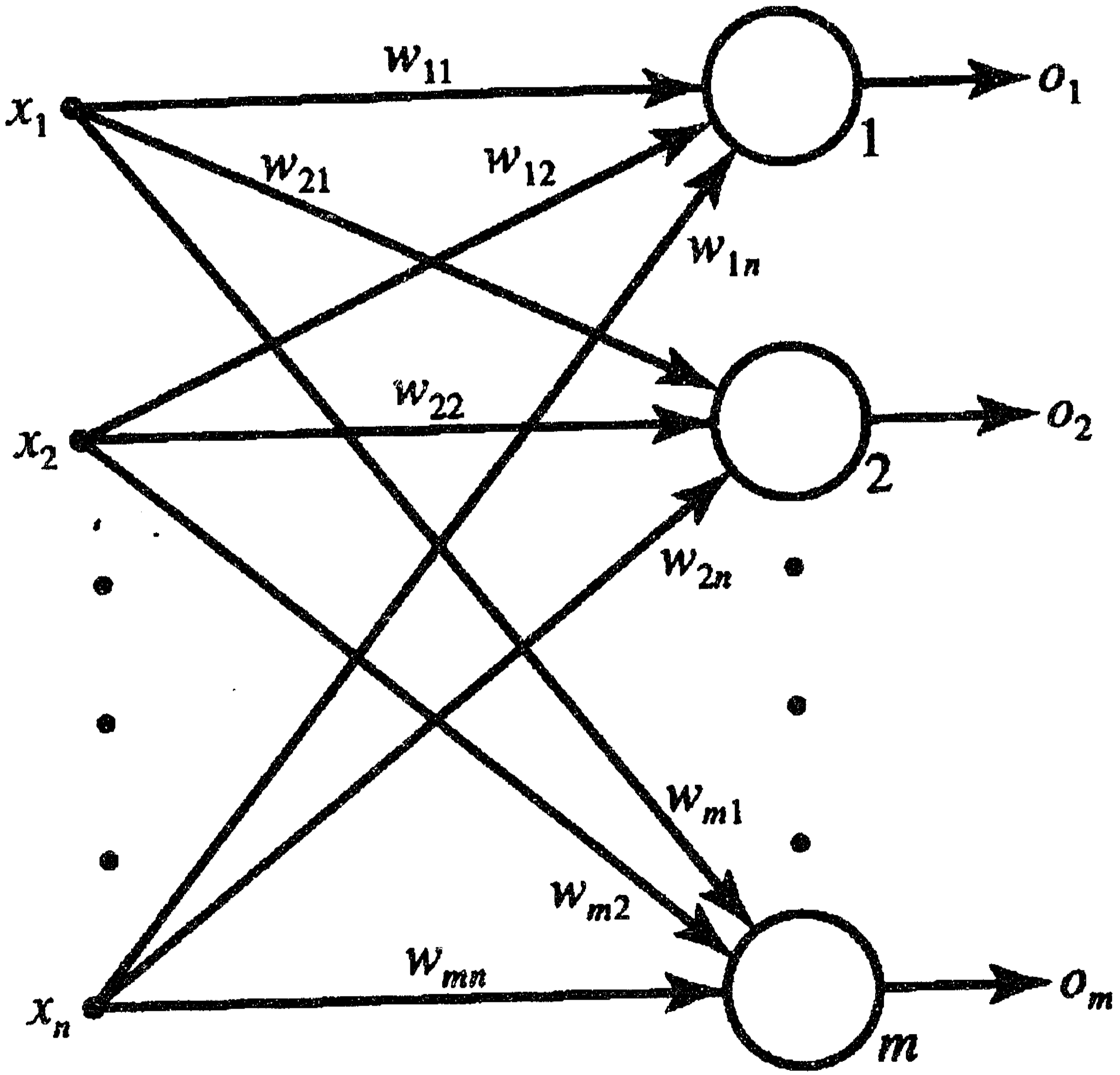


Figure 2: Feed Forward Neural Network Architecture. The input values X_i are on the left. Each input is connected to each node, or neuron, through a weighted link. Each node sums up the incoming weighted inputs, passes that sum through a simple nonlinear function, and then outputs that value to the next layer of nodes, represented by the O_i on the right. The network learns by comparing the predicted output value to the actual correct output value. The values of the weights, W_{ij}, are then adjusted to reduce the difference between the predicted and actual output values.

is subtracted from the predicted value, yielding a measure of how far wrong the network's prediction is. The squared value of this difference is known as the error. The network then uses a learning algorithm, such as backpropagation, to make small adjustments to all the weights of the network in order to reduce the network's error for that data point, or example. With back-propagation, these small adjustments are computed using gradient descent, which means that the network computes the changes to be made based on the derivative of the error function with respect to each weight. In effect, this process adjusts the weights of the network in the direction that reduces the error the most (Rumelhart 1986).

This process is repeated hundreds of times in training a network. One cycle through all the training examples is known as an epoch, and it is not uncommon to train for hundreds of epochs. Deciding when to stop training is best accomplished by having a second dataset, known as the testing or validation set, which the neural network does not use in training. After every training epoch, the error over the entire test set can be computed while holding the network weight values fixed. As the network trains, error on this test set usually decreases for awhile, but eventually the error on the testing set begins to go up even as the error on the training set continues to decrease. This phenomenon is known as overfitting, and is a result of the network memorizing the specific examples of the training set instead of learning the general underlying pattern that the data represents. Overfitting causes the network to fail to generalize well to new data. Because generalizing to new data is the ultimate goal in building a model, training is stopped when the Mean Squared Error (MSE) on the testing set stops decreasing from one epoch to the next (Pal 1992).

IV. Data and Data Preprocessing

Dataset Characteristics

This section describes the data used in the modeling efforts. For prediction of Hot Metal Temperature (HMT), a data point consisted of 35 input variables plus a measurement of HMT. Data were recorded during the steel making process at 5 minute intervals. A list of variables is included in Table 1 on the next page.

Because of the real world environment from which the data were collected, measurements on each variable for each 5 minute period were not always possible to obtain. If the value of a variable was missing, then that value would be filled in with the last previous known value. For most inputs this was rare, but for some inputs frequent measurements are costly and undesirable, so many values were filled in. Although not ideal, constraints such as this are unavoidable in real world applications of neural network techniques.

This had a tremendous effect on the target output variable, Hot Metal Temperature. Because of the extreme conditions inside the blast furnace, it is practically impossible to get measurements of HMT from inside the blast furnace. Therefore, HMT is only measured when the hot metal is tapped from the furnace. Tapping is an important operation which is prohibitively expensive to perform solely for the purposes of taking a measurement. Rather, the tapping operation is performed only when there is a significant amount of metal to be tapped and not otherwise. This usually occurs approximately every 45 to 75 minutes, with the average time between measurements being about an hour.

RAFT temperature
Group 2 heat flux
Group 1 heat flux
Steam
Actual Coal injection (kg/thm)
Hydrogen (from top gas analysis)
Central working index
Total quartz
Differential pressure
Total limestone
Wind volume
Coke moisture from bin #9
Top gas % hydrogen
Total dolo
Coke moisture from bin #7
Hot blast temperature (deg. C)
% carbon dioxide

Top pressure (kg/sq. meters)
Ore/Coke ratio
Hot blast pressure
Cold blast pressure
Charge time for 10 semi-charges
Total Ore Charged
% Sinter in Burden
Total sinter
Total nut coke
% Oxygen Enrichment
Total coke
Carbon oxide (from top gas analysis)
Permeability Index, K
% carbon oxide
Total Ti-ore
Theoretical carbon rate in kg/thm

Table 1: A list of the original 35 input variables

This presented a serious problem for the modeling effort. Although there were on the order of 8000 total data points, only about 1 in 12 of these points had a correct value for the target variable. This left the project team in the unenviable position of trying to accurately model and predict the value of a variable which was only accurate for 8% of the data points. For reasons such as this, preprocessing steps would be necessary in order to improve the performance of the models.

Preprocessing of Data

The raw data from the blast furnace operation were unsuitable for direct use. Reasons for this ranged from problems inherent to the data, such as missing or very anomalous values, to more subtle flaws, such as not taking into account the effect of time lags in the production process. Several steps were involved in processing the raw data into a dataset that would be well suited for training an artificial neural network.

Normalization

Neural networks, along with many other statistical learning algorithms, are sensitive to the scale and variance of the input variables. For this reason, normalizing of variables is often performed. The data were preprocessed by normalizing all variables between 0 and 1. This gave all of our variables the same range of values, along with relatively similar means and variances. These methods allow the neural network to capture more easily the relationship between a change in inputs and the change in the output, avoiding distortion caused by wide differences in scale and variation.

Linear Interpolation

A major problem with the original data were inaccurate values of HMT for many of the data points. As described above, HMT is only measured approximately every hour, but data points were taken every five minutes. As described above, when the true value of HMT was missing, the last known measured value, which came from the most recent hot metal tapping, was filled in. This caused the graph of HMT over time to be a step function, which stayed constant for a period until jumping to a new value when a new tap was taken.

Clearly temperature does not behave like a step function. A more accurate model of temperature change would incorporate the knowledge that it is a gradual process. Therefore, linear interpolation between measurements of HMT was used to approximate values for the missing data points. Thus, the data with unknown HMT values are improved by providing a better estimate of what the true HMT value would be.

Linear interpolation is not the only interpolation method available. Alternative models of interpolating were considered, such as logarithmic or exponential decay, but with virtually no knowledge of the process of temperature change there is no reason to pick any other alternative interpolating function. In this situation, a linear interpolating function will yield estimated values with the lowest expected squared error from the actual HMT.

Hourly Averaging

The raw data from the blast furnace contained data points taken at 5 minute intervals. Early modeling efforts, by default, used the data at this 5 minute level time scale. However, insights into the blast furnace process led the project team to believe that perhaps a larger time scale might produce greater predictive ability. This is because the five minute level data may see some inputs swinging rapidly from one value to another, but the temperature only changes slowly over a longer period of time. Thus, these short term changes do not have a noticeable effect on the output and so the ANN learns that these inputs are not important in predicting HMT. Conversely, longer term trends in the input variables do have an effect on the Hot Metal Temperature. Domain knowledge from experts at TISCO indicated that an effective unit of time for considering the data would be in blocks of one hour.

Implementing this knowledge, groups of 12 data points were averaged to create one data point which represented an hour block. Preprocessing the data in this way increased the accuracy of the model, as can be seen in the Results chapter.

Moving Window Averaging

While hourly averaging of the data improved the predictive ability of the network, it had a side effect of greatly reducing the number of data points available for training the networks. However, because the data was originally in five minute intervals, the hourly averaging of data points was not making the most effective use of the available data. A method to use the information contained in the five minute interval data but with the advantages of the hourly averaging was needed. The project team implemented such a method, known as a moving window. The moving window method begins by taking the first 12 data points and averaging them, the same as the hourly averaging method described above. The difference is in the next step, where the moving window shifts over by a five minute interval, or one data point, and then averages the new data point with the previous 11 points. The window then continues to slide one data point at a time, until the end of the data is reached. Thus, the resulting dataset has only 11 fewer data points than the original dataset. This is clearly better than the hourly averaging technique which reduced the data set by a factor of 12.

The techniques used for data preprocessing proved to be a surprisingly important determinant of how easily and accurately the neural network could learn the mapping represented by a given dataset. The importance of this preprocessing will be discussed further in the Results chapter.

The data processing steps described above were implemented as Matlab functions (Matlab is a widely used program among engineers and scientists for doing computationally intensive tasks). These preprocessing functions are a valuable addition to the NNRUN neural network software package, which is the topic of the following chapter.

V. NNRUN: A Neural Network Software Package

NNRUN Package Overview

The NNRUN system is a powerful Artificial Neural Network (ANN) time-series analysis package developed on the basis of Matlab's NNET toolbox . It was developed to help automate the search for the optimal neural network configuration for a given dataset. Credit for the development of the NNRUN package is shared with Vladislav Gabovich, a member of the MIT/Sloan datamining group.

NNRUN has a menu-driven user interface that encompasses all of the package's modules, including the primary ANN training/testing tool (NNTESTER) and a prediction module (NNPREDICTOR). The package provides the user with the ability to:

(1) Build an ANN designed to model a specific time-series analysis problem

(2) Design the ANN from a wide range of configurations

(3) Configure the ANN to satisfy any performance/computing power requirements

(4) Select any combination of ANN training parameters

(5) Automatically optimize the ANN weight/layer structure

(6) Let NNTESTER select the best possible ANN for the modeled system

(7) Simultaneously train and select from any number of ANNs

(8) View 2- and 3-dimensional graphical representation of relative ANN configuration performance, as well as graphing the performance of individual networks

9) Retain complete information about all tested ANNs

(10) Use NNPREDICTOR to simulate output on any set of input data using previously trained ANNs

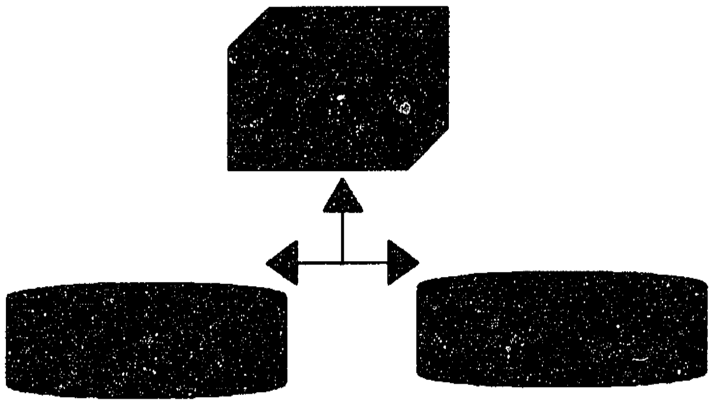


Figure 3: NNRUN Basic Software Package Architecture. NNRUN is the user interface to the package, NNTESTER is the neural network based modeling engine, and NNPREDICTOR uses models produced by NNTESTER to predict values for new data.

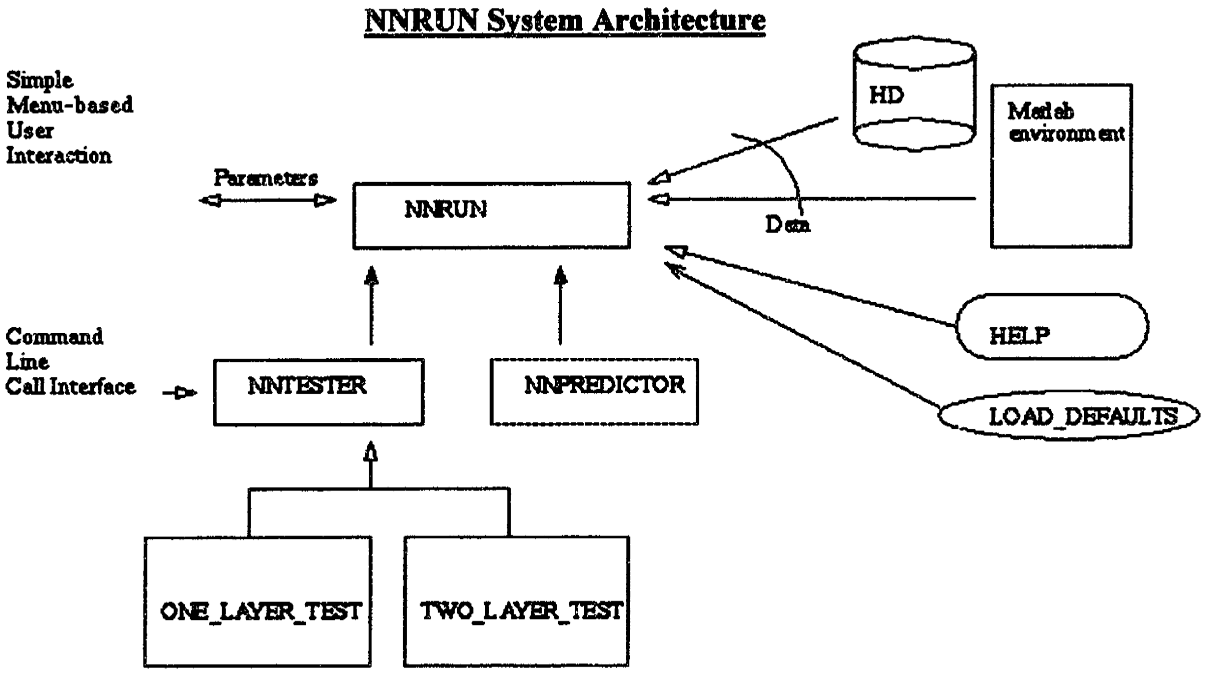


Figure 4: NNRUN System Architecture. A more detailed look at the components and interactions of the NNRUN software package. This figure also shows input and output relationships to the external environment in which NNRUN is being used.

NNTESTER Overview

The main task of the NNTESTER component of NNRUN is the modeling of a given input to output mapping using neural network models. Specifically, given a 'many-to-one' discrete mapping, NNTESTER creates multiple Neural Network (NN) objects whose purpose is to model, to a high degree of accuracy, any functional mapping, including nonlinear relationships.

In the analysis of time series data, the NN objects which have been found to accurately model the mapping, as judged by performance on historical data, can then be used to predict future values. The unique feature of the NNTESTER package is the dual benefit of the complete automation of the process of building the optimal NN model of the given mapping, selecting the best-performing NN from an extensive set of NN configurations, together with the provision of a highly intuitive interface that allows the user to access and modify all relevant system parameters to tailor NNTESTER to the users specific needs. The high level of automation eliminates the need for human AI knowledge engineers, bringing the power of neural network modeling to general users, while the easy accessibility of system parameters provides room for extensive customization of the package to the needs of any professional.

NNTESTER Architecture

Overview

In order to automate the selection of the neural network with the best predictive powers, and to ensure that the error measure of NNs' performance accurately represents their ability to model the mapping, NNTESTER relies on two main internal loops:

(1) The Data loop, controlled by NUM_RUNS parameter, and

(2) The Configurations loop, controlled by the RANGE_NODES parameter.

Data loop: partitions, default and mean performance

The original version of NNTESTER measured the performance of a particular neural network by recording the error between the predicted and target output over the 'default' training/testing data partition. That is, the last TEST_FRAC of the data points were reserved for testing, and the first (1-TEST_FRAC) points were used to train the neural networks, in accordance with chronological ordering of the data points.

Because the number of data points may be small, if performance of various neural nets is compared using only one particular partition of data into a training and testing sets, the best performing network will necessarily be the one that is best fit to describe the particular data subset, and not necessarily one that best describes the overall mapping (see Data Preprocessing). To alleviate this problem, the user has the option to set the NUM_RUNS parameter. This parameter allows the user to train the neural networks separately on a large number (up to the number of data points) of different testing/training partitions of the original, resulting in the selection of the NN configuration with the best mean performance over different partitions, hence ensuring that the "best" configuration reflects the general properties of the mapping and not a particular choice of data (partition). Another parameter, RAND_RUNS, specifies whether the different partitions are selected randomly or sequentially from the main data set. Based on the results of running NNTESTER with different number of runs, it was confirmed that the ANN which exhibits the lowest error on the default data partition rarely provides the best mean-performance NN.

Further, we observed that the mean-squared error (MSE) indicators of performance are significantly higher for the default partition than the MSE errors for randomized train/test selections. It is probable that this discrepancy is caused by fundamental changes in the properties of the mapping in the time period enveloping the train/test partition.

Configuration loop: Automation

The RANGE_NODES parameter in NNTESTER (or appropriate choice in Configurations Menu of NNRUN) allows the user to select a range of NN configurations, specifically either a two-layered or a three-layered NN with a specified range of the number of nodes in each layer. Each of these configurations is then initialized, trained and tested by NNTESTER, and the networks with both best mean and optimal default performance are identified, and returned.

In combination with the other training parameters, the capability of defining the field of search, allows the user to train a large class of ANNs under relaxed training performance conditions in order to identify the general configuration type that seems to exhibit the best predictive power, and then constrict both the number of tested configurations and the training performance constraints, "zooming in" onto the best configuration. We found this technique to be particularly helpful in the analysis of all blast furnace output variables.

NNPREDICTOR

The NNPREDICTOR module is the natural complement to the NNTESTER package. Using a previously trained ANN and an input dataset, the NNPREDICTOR makes predictions for the output variables based on the mapping the ANN learned in the process of training with NNTESTER.

The integrated data preprocessing tools allow NNPREDICTOR and NNTESTER to be used together seamlessly. When a dataset is processed for training use by NNTESTER, the parameters necessary to process future data in exactly the same way are stored. Then, when the user wants to predict new data, the preprocessing utilities are passed the stored information to generate a dataset with the same processing methods used.

The NNPREDICTOR is a completely stand-alone module, however, and can be directly used outside the NNRUN package for prediction purposes. NNPREDICTOR enables the user to load an ANN from any Matlab workspace file, run the prediction algorithm, and then view, discard, or save the prediction results.

VI. Results

This chapter describes results obtained from training artificial neural networks on data obtained from the TISCO blast furnace. The first section covers results of modeling hot metal temperature (HMT), and the second covers predicting the silicon content of the pig iron produced by the blast furnace. The subsections describe the results from modeling data sets which were preprocessed in several different ways.

Modeling Hot Metal Temperature

Results from using original data

The initial results from the neural network modeling were not very promising. Due to time-dependency problems and problems regarding the frequency at which the hot metal temperature was measured (see previous section for details), the predictive power of the network was severely hampered. The result of running a feed forward network on the data using the 10 input variables that were the most correlated with hot metal temperature (without using time lags and without linear interpolation of the hot-metal temperature) is shown on the next page.

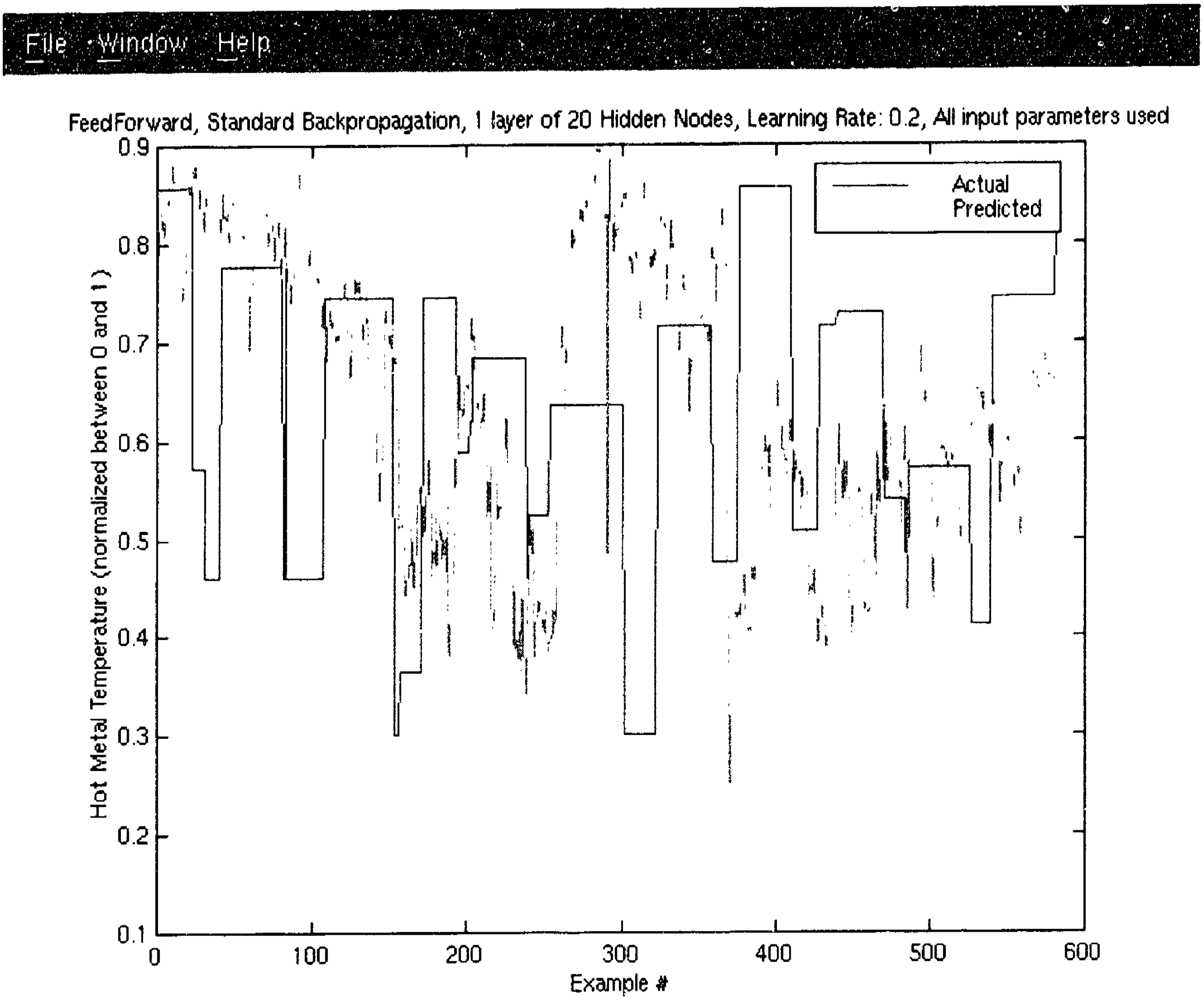


Figure 5: The results of a feed-forward neural network trained with standard backpropagation on the original data. The line which is a series of step functions is the actual HMT. The jagged line is the predicted HMT.

Including previously measured value of HMT as an input

In correspondence, Dr. D. Bhattacharjee of the TISCO (Tata Iron and Steel Co.) blast furnace suggested using recent measurements of HMT in order to predict the current value. Although the current hot metal temperature was the linearly interpolated value for each example, the previous hot metal temperature used as an input variable for that example was not the previous linearly interpolated value. Instead, it was the actual last measured value. Since the linearly interpolated value at time t-1 is so close to the linearly interpolated value at time t (this is a characteristic of the linear interpolation), the system would be cheating if it had the previous, linearly interpolated hot metal temperature value when predicting the current hot metal temperature. The dataset was also expanded using the moving-window technique described in the previous section. The results of running a feed forward network (with standard back propagation and 10 hidden nodes) using the other 35 input values plus the last previously measured HMT value appears on the next page.

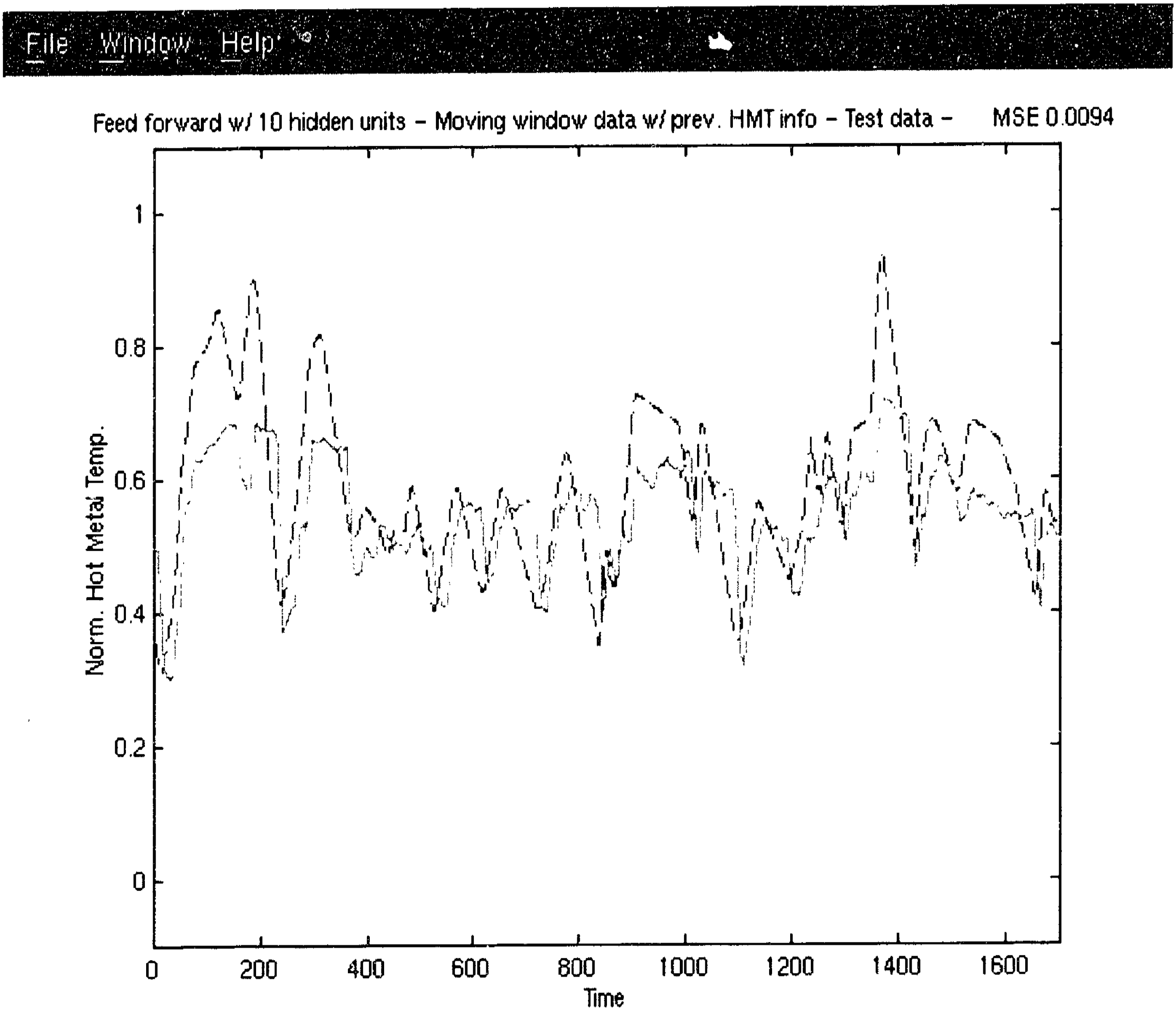


Figure 6: The result of using a feed forward neural network on data that includes the previous hot metal temperature as an input to the model. (The dashed line represents the actual HMT and the filled line represents the predicted HMT)

Using Prior Knowledge to Reduce Number of Input Variables

Further analysis of the data revealed that some of the input variables were redundant, and some others were not useful in predicting HMT. While in theory neural networks are capable of ignoring redundant and/or irrelevant variables, this property requires massive amounts of data to hold true (Lippman 1997). With the constraint of a finite amount of data, the network will more reliably capture the mapping from inputs to outputs if only relevant input variables are considered.

Through analysis of the data and correspondence with blast furnace experts from TISCO, the number of input variables was narrowed from 35 to 11. These 11 variables were: total coke, carbon oxide (from top gas analysis), hydrogen (from top gas analysis), steam (tons/hr), group 1 heat flux, group 2 heat flux, actual coal injection, % oxygen enrichment, ore/coke ratio, hot blast temperature (degrees C) and charge time for 10 semi charges.

To test whether these 11 inputs could capture the same information that was contained in the complete 35 variable data set, a total of 8 different cases were examined. Feed-forward networks using standard back-propagation and 10 hidden nodes were run on the 11 variable data, both with and without the previous measured HMT as an input. Similar networks were then run on the set of 35 input variables, both with and without the use of the previous measured HMT as an input. These same 4 conditions were then repeated using a standard linear regression model. The mean squared error results (calculated using the testing data) is displayed on the following page.

Feed-Forward Networks using Standard Back-Propagation and 10 hidden Nodes

	Mean Squared Error on Training Data	Mean Squared Error on Testing Data
11 Variable data without previous HMT as an input	0.0187	0.0164
35 Variable data without previous HMT as an input	0.0187	0.0160
11 Variable data with previous HMT as an input	0.0055	0.0083
35 Variable data with previous HMT as an input	0.0044	0.0075

Linear Regression

	Mean Squared Error on Training Data	Mean Squared Error on Testing Data
11 Variable data without previous HMT as an input	0.0116	0.0292
35 Variable data without previous HMT as an input	0.0089	0.0183
11 Variable data with previous HMT as an input	0.0044	0.0086
35 Variable data with previous HMT as an input	0.0040	0.0074

Table 2: Comparison of linear regression models and feed forward neural network models using 11 variable and 35 variable datasets, both with and without including previous measurements of HMT. Comparisons based on testing error are most meaningful, though training error is included for completeness.

Predicting Future Values of Hot Metal Temperature

The preceding sections have shown promising results in predicting the current Hot Metal Temperature (HMT). However, predicting the current HMT is of limited value. A much more useful model would predict the HMT anywhere from a few minutes in advance to many hours into the future. In this section, results are presented of modeling HMT 1 hour, 2 hours, 4 hours, and 8 hours into the future.

Of course, trying to predict into the future also introduces more inherently unpredictable noise. As forecasts are made further into the future, there are several sources of noise that add to the unpredictability of the process. These sources include operator intervention, changing dynamics of the process, effects of unmeasured variables. By far the most important of these sources is the operator intervention. Because of this additional noise, the team expected a drop off in accuracy as the length of prediction time grows.

The 11 input variables that have been shown to be most strongly correlated with output data, along with the current and previous value of HMT, were used to predict HMT into the future. Hourly averages of the data were taken to reduce the effects of random noise and small-scale variations in the input parameters. However, moving window averaging was not used as the average of the data in a window incorporates a significant amount of knowledge about the value of the data at the end-points of the window. This would have provided the ANN with more information about the future values of HMT, than it could be expected to have in a real-life situation. The current value and the last known value of the HMT are used for making future predictions. In further experiments, it might be worthwhile to try using as input known values of HMT even further back in time.

The first graph presented shows prediction of the current value of HMT using the 11 variable data described earlier along with the two previous values of HMT. This graph serves as a comparison for the results presented in the following pages, which show forecasting of HMT at increasing periods of time into the future.

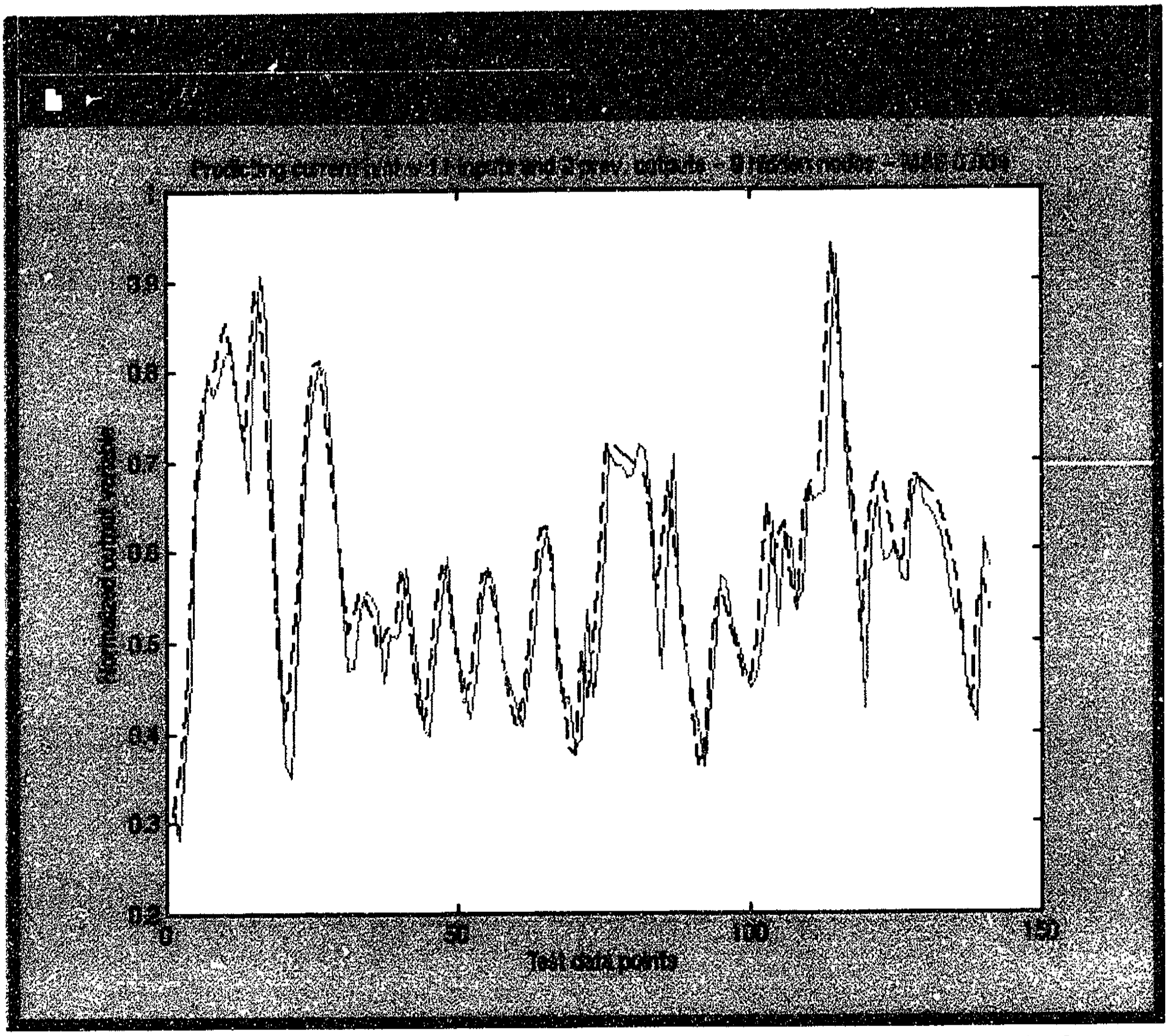


Figure 7: Prediction of current HMT using 11 variable data along with 2 previous values of HMT. Compare results here to following graphs of prediction into the future. In all following figures, dashed line represents actual values of HMT, solid line is the predicted values of HMT for the same time period.

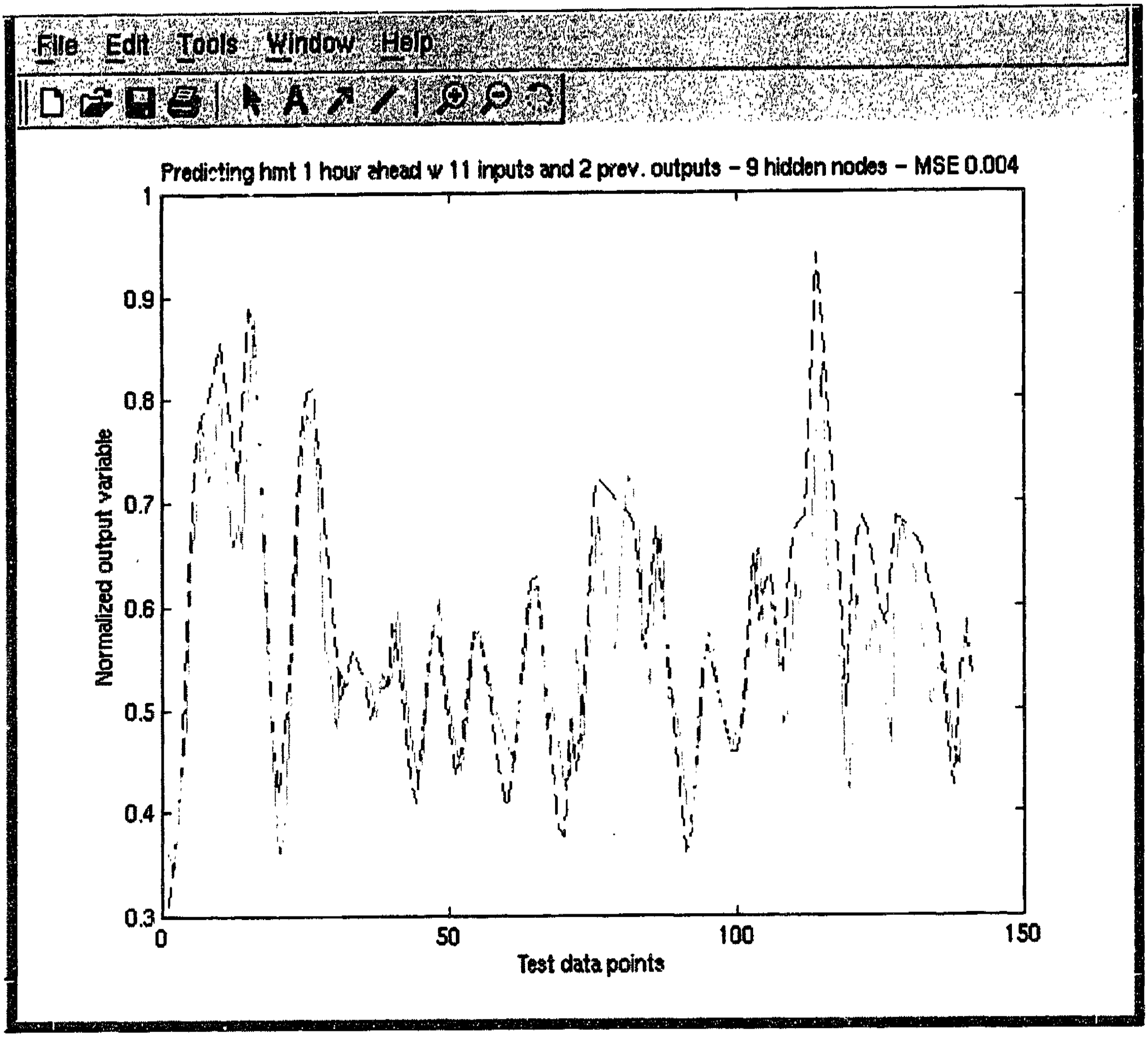


Figure 8: Prediction of HMT one hour into the future. Notice that there is no general pattern of lagging or leading, indicating the model is doing more than just predicting the last known value. At this stage, predictive accuracy is still excellent. The accuracy of this model and the previous model are nearly equal (both have MSE of 0.004).

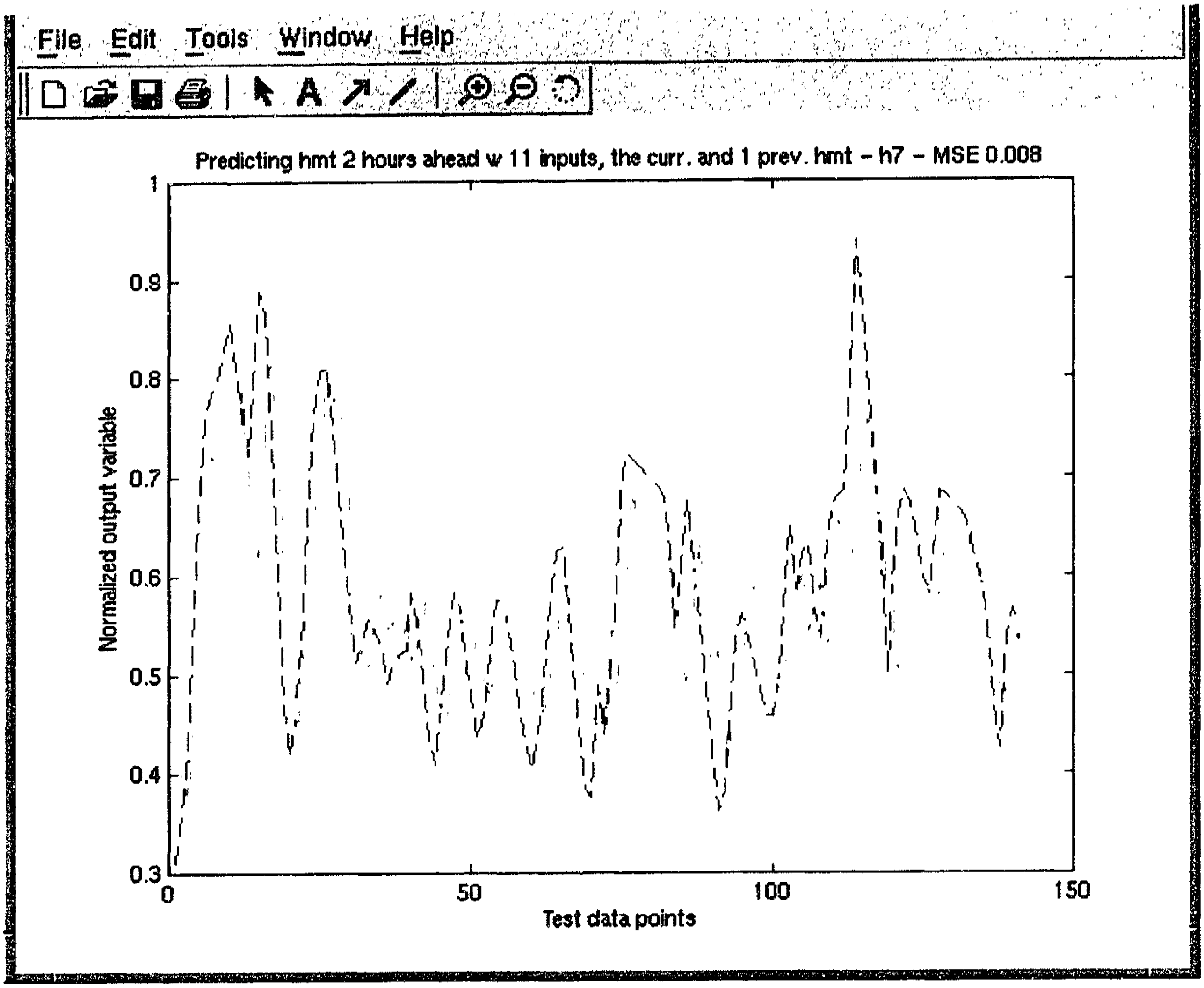

File
Edit
Tools
Window
Help
Predicting hmt 2 hours ahead w 11 inputs, the curr. and 1 prev. hmt - h7 - MSE 0.008
Normalized output variable
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
50
100
150
Test data points

Figure 9: Prediction of HMT two hours into the future. Here the model's prediction begins to lag the true value (the predicted curve appears shifted to the right of the actual value). This indicates the model is primarily relying on the current and previous HMT values to make a prediction for what the HMT will be two hours in the future.

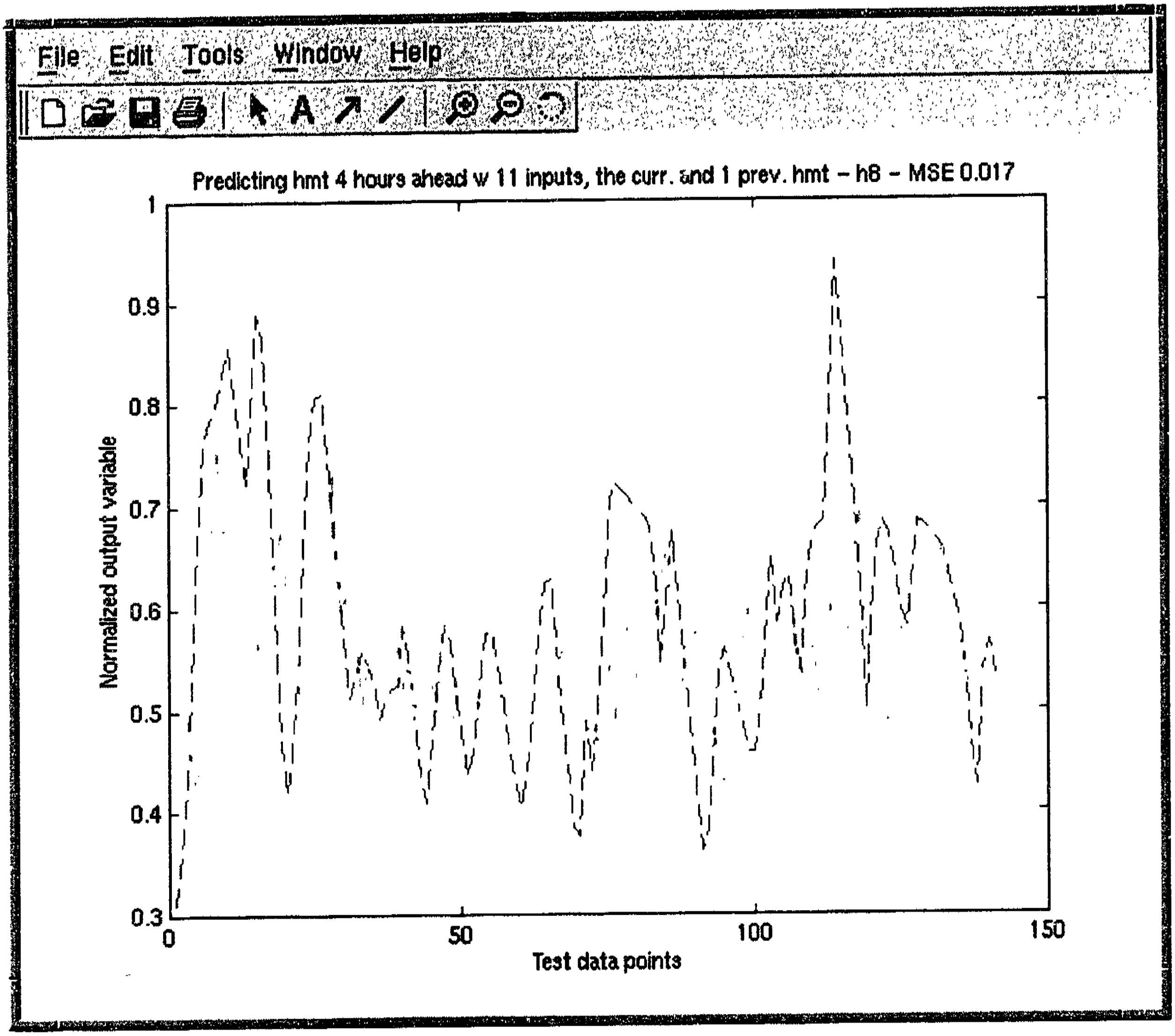


Figure 10: Predicting HMT four hours in advance. Results here continue to become less accurate, though the model still predicts much more accurately than a baseline prediction of the mean value. The prediction is noticably lagging the actual value (the solid line, prediction, appears to be shifted to the right from the actual value). This indicates the model is relying heavily on the previous values of HMT used as inputs.

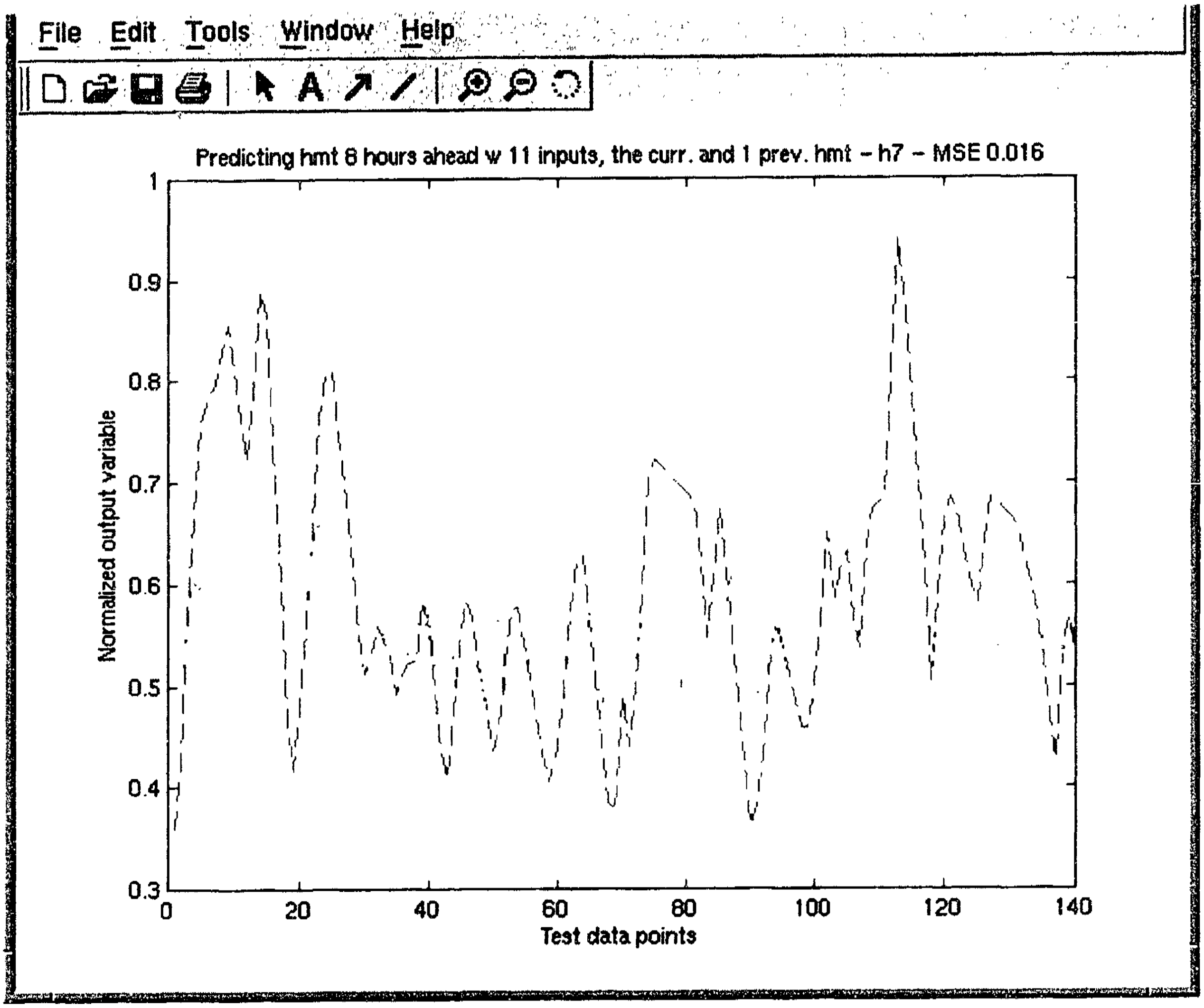

File Edit Tools Window Help
Predicting hmt 8 hours ahead w 11 inputs, the curr. and 1 prev. hmt - h7 - MSE 0.016
Normalized output variable
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100
120
140
Test data points

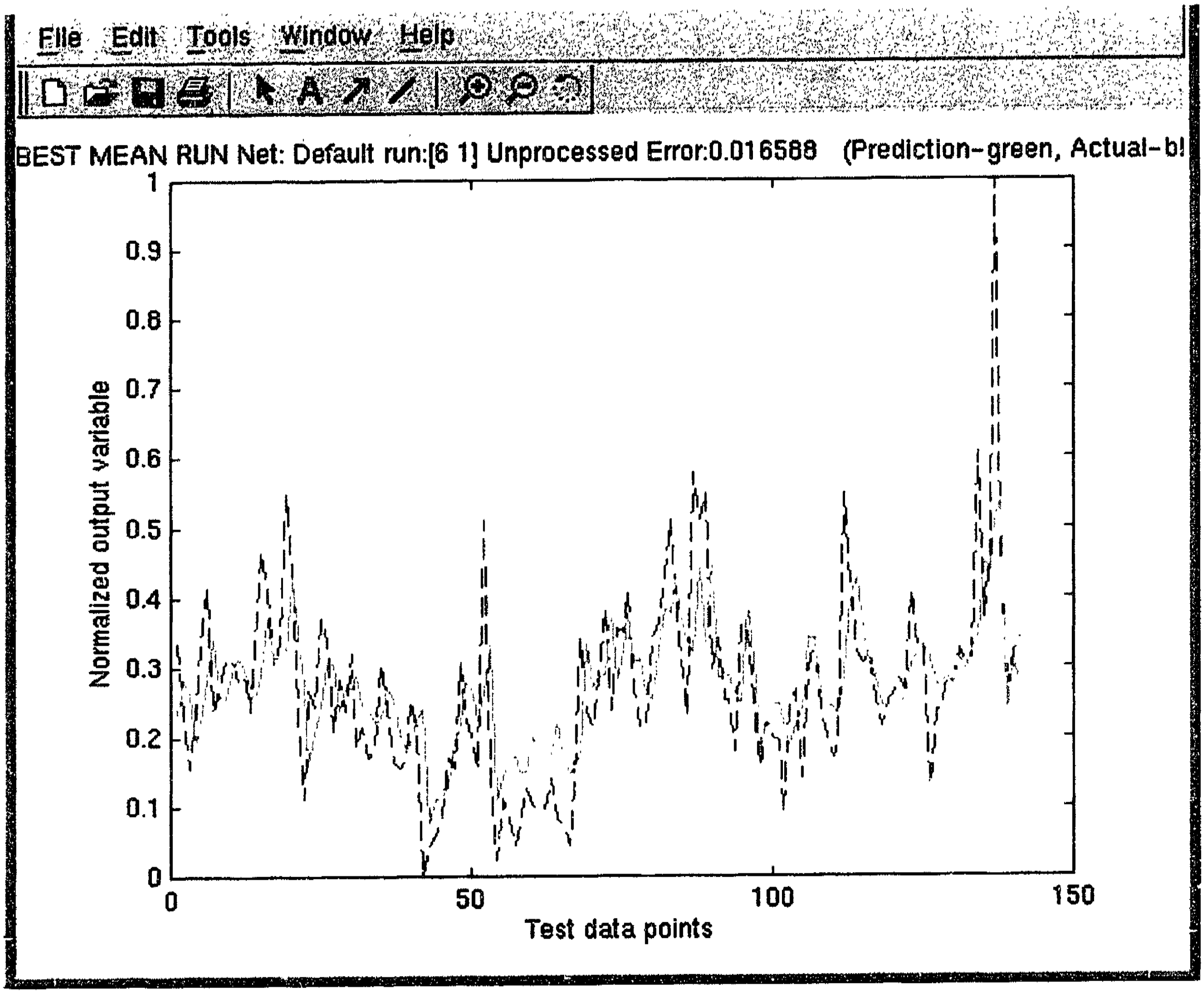
Figure 11: Prediction of HMT 8 hours into the future. This far into the future, the introduced noise makes very accurate predictions impossible. Performance is drastically reduced from the one hour prediction interval. However, the model is still predicting better than a simple mean value prediction.

Prediction and Modeling of Hot Metal Silicon Content

An important variable to control in producing pig iron for steel production is the silicon content. Silicon is an impurity which causes steel to be brittle and reduces the load bearing properties that make steel useful in construction (Biswas 1984). In producing steel, there are several methods which can be used to remove silicon from the pig iron, but these methods are expensive. It is much better if the silicon content of the pig iron can be kept low. Normal ranges for silicon content are 0.4% - 0.8%; silicon content above 0.8% necessitates expensive measures to reduce the silicon content later in the steel production process.

The first step in being able to control the silicon content of pig iron is to be able to predict what the silicon content will be. Towards this goal, the research team used NNRUN and associated data processing tools to model and predict silicon content of historical data. The input variables were the same variables used in predicting Hot Metal Temperature; these are described in the chapter "Data and Data Preprocessing". The HMT was also used as an input variable in predicting silicon content. The output variable, silicon content percentage, was measured by chemical analysis of samples from each tap of the hot metal. In predicting the next value of silicon content, the last two known values of the silicon content from previous taps were used.

The figures on the following pages show results of predicting the next value of silicon content based on the above dataset.



BEST MEAN RUN Net: Default run:[6 1] Unprocessed Error:0.016588 (Prediction-green, Actual-b!

Figure 12: Prediction of Hot Metal Silicon Content. Predicted values track quite closely with the actual values. Prediction is for the next tap, approximately 90 minutes into the future. The network used has 1 hidden layer of 6 nodes. Dashed line is actual value, solid line is the network's predicted value

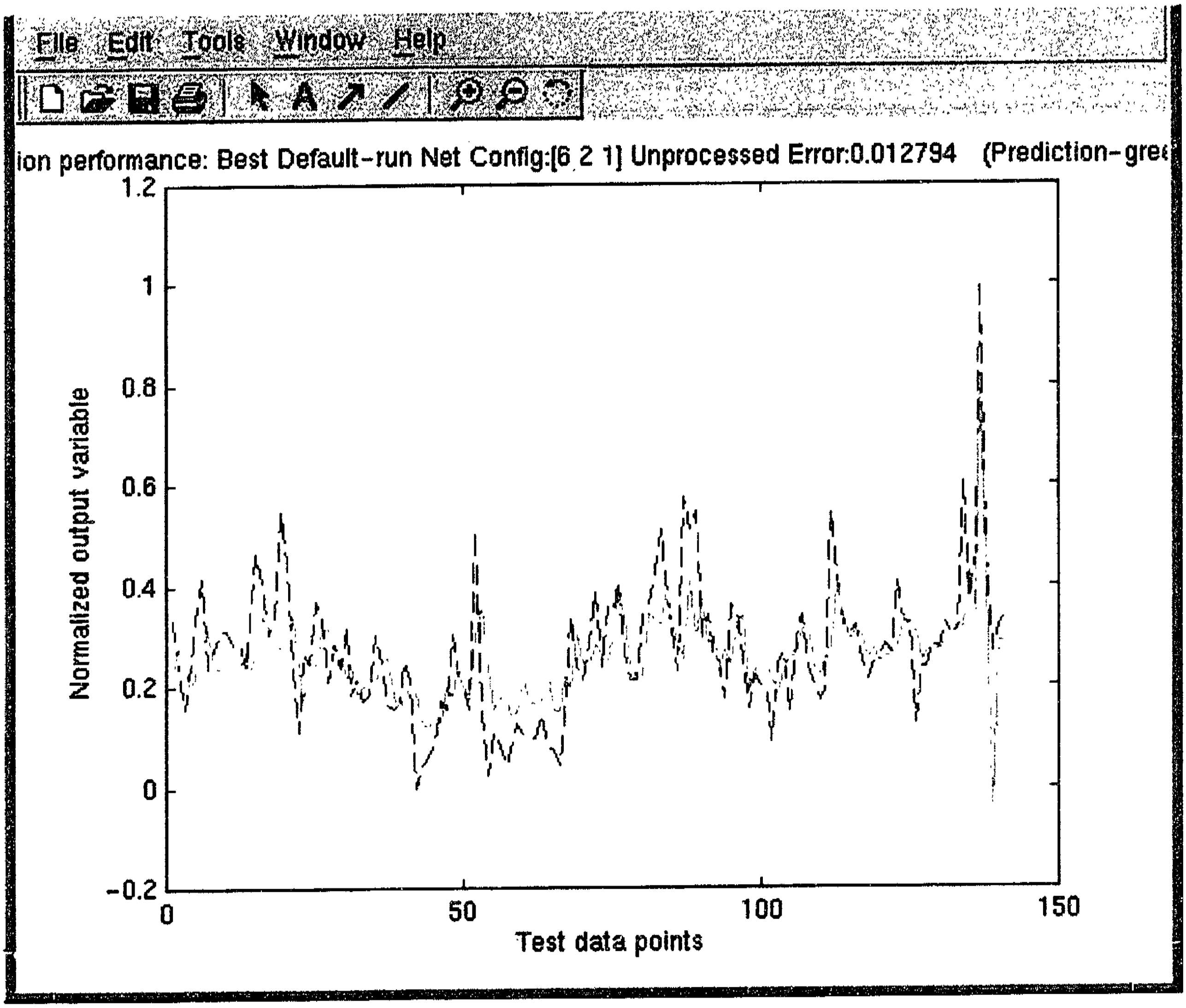

File Edit Tools Window Help
ion performance: Best Default-run Net Config:[6 2 1] Unprocessed Error:0.012794 (Prediction-gre
Normalized output variable
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
50
100
150
Test data points

Figure 13: Prediction of Silicon Content with a two hidden layer network. Performance is slightly better, but still quite similar, to the accuracy achieved by the much simpler one hidden layer network of six nodes presented in Figure 12.

Discussion of Results

The results presented here show that modeling and predicting blast furnace operations can be done to a high degree of accuracy. The combination of appropriate data preprocessing methods and the NNRUN system have resulted in models that, on average, account for 95% of the variance in Hot Metal Temperature and Hot Metal Silicon Content.

Results of Predicting Hot Metal Temperature

When using the previous HMT value as an input, a feed-forward network (with 10 hidden nodes) was able to attain a mean squared error on the test data of 0.0075. This was the best performance seen so far.

When the previous value of HMT is not used as an input, the neural networks are essentially just predicting around the mean of the HMT values. Thus, it does not matter much if 11 or 35 variables are included, because the network learns that the best prediction is right around the average of the data. It is a bit strange that the linear model, in the case when previous HMT is not used, shows somewhat better results than the neural networks. The explanation probably lies in the fact that the linear regression model produces the optimal linear solution, while the neural networks could be converging on solutions that are local minima in the error space of the system. Therefore, the neural net just predicts near the average value.

In the case where previously measured HMT is used as an input, the overall results turned out to be much better. Both the neural net and linear models converged to the same solution. This could mean that the ANN models are capturing all of the useful information in the data, while avoiding overfitting to the noise. The increase in the quality of the results seems to be directly attributable to using the previous HMT, indicating that interactions between previous HMT and the other variables may not be important.

Results of Predicting Hot Metal Silicon Content

Using neural networks to predict silicon content was quite successful. A single hidden layer network of six hidden nodes, using two previous values of silicon content along with other input variables, resulted in predictions that had a MSE of 0.017. This result, when presented to steel production experts, was considered very good and exceeded their expectations of how accurately silicon content could be predicted.

Initial attempts which did not incorporate hot metal temperature as an input nor previous values of silicon content were much less accurate. These models performed only as well as a simple linear regression model. Again, as in the case of predicting HMT, improved preprocessing of the data resulted in a dramatic increase in predictive accuracy. The fact that this increase occurs when HMT and previous silicon content values are included indicates that these are important variables in predicting future values of Hot Metal Silicon Content.

The best results obtained came with a network architecture of two layers of hidden nodes, the first layer having six nodes and the second two nodes. This model had a MSE of 0.013. While this is a lower error than the single layer network discussed above, it is important to note how much more complex this model is. While the more complex model did achieve better results, the difference is slight compared to the increase in complexity. This increase in complexity means increased time spent in training and, especially time consuming, searching through the space of possible models. Without the NNRUN system, such a search through model space would be infeasible for a human operator to conduct.

VII. Conclusion

Neural networks can successfully be used to model complex industrial processes. While traditional non-data driven techniques suffer from a lack of flexibility and limited accuracy, neural network techniques can learn complex, nonlinear relationships. This property allows such networks to learn a mapping from the input variables, such as characteristics of raw materials, control settings, and sensor measurements, to output variables such as hot metal temperature and silicon percentage.

A surprising result which came out of this work is the critical role importance of preprocessing the data. Appropriate preprocessing allows the network to learn more accurately the relationship between inputs and output. These steps can bring buried patterns contained in the raw data closer to the surface, where the neural network can more easily capture them. A running theme in this work was the dramatic improvement in model accuracy as the methods used to preprocess the data became increasingly appropriate and sophisticated.

Another interesting insight was that different network architectures and learning algorithms were not significantly different. While some types of network and algorithms were not suited for this application, among those that were believed to be well suited there was not much variation in performance. Particularly, the best performing architectures and learning algorithms were all very similar in their predictive accuracy levels. This supports the view that the networks are capturing most of the predictive information, while the residual error is likely to be due to unpredictable noise. However, it is impossible to say with certainty that alternative models could not perform better.

These two insights, taken together, indicate that the most important work in modeling complex industrial processes is in the preprocessing of the data, and not in deciding which network architecture or learning algorithm to use. Normalizing the data, extracting important features, taking into account time lags for variables, finding the right time scale to use, efficiently using all data available, and incorporating previous values of the output all individually helped improve the accuracy of our model. Taken as a whole, the effects of intelligent and appropriate preprocessing are quite striking.

A possible direction for future research in this area would involve comparing other types of statistical learning machines, such as genetic algorithms, ID trees, and/or support vector machines. Given the lack of differentiation between the different types of neural networks, it would be a surprise if any of these methods performed significantly better than the neural network models used here. Still, it remains to be seen whether this hypothesis would in fact be correct.

Another possible research topic would be to experiment with more forms of preprocessing. A few options include using more complex normalizing methods, such as Independent Components Analysis (Karhunen 96), reusing input variables with a different time lags, and obtaining expert information about possible combinations of variables that might be added to the dataset as a separate variable. There are certainly more forms of preprocessing that could be explored in addition to the ones mentioned here. With the improvements in model accuracy and prediction seen through preprocessing in this study, there is every reason to believe that some of these methods will significantly improve on the results presented here.

The most important direction for future research, however, would be to take the *predictions* of models presented here and use them to *control* the process. While prediction alone does have its uses, ultimately the value in making a prediction is the ability to use that prediction to produce a better outcome than the one currently predicted. For example, in pig iron production, a goal of the blast furnace operator is to keep the Hot Metal Temperature within an acceptable range. Too low, and the quality of the hot metal produced is low; too high means the operator is wasting expensive resources and operating inefficiently. The work presented here allows the model to predict that HMT is going to be outside of the band. But given that prediction, what should the operator do? What is the optimal way to control the blast furnace inputs to bring the predicted temperature back within the acceptable range? This is an open question that the research group intends to pursue next.

This thesis has presented a case study in applying neural network techniques to the modeling and data mining of blast furnace operations. NNRUN, the neural network software package, was developed to aid in this effort. As valuable as NNRUN was to the research presented here, it will be even more valuable as a general modeling tool that can be applied to any problem where neural networks are used. The NNRUN system features the ability to automatically search through a wide range of potential neural network configurations to find the optimal ANN for a problem. This feature makes NNRUN useful for many modeling tasks.

As shown by the results, neural network methods have successfully been able to predict both the hot metal temperature and silicon content of pig iron produced in a blast furnace. This is a promising result for potential applications of neural network techniques to other complex industrial processes. In the area of blast furnace operation, the results of this work will help produce higher quality pig iron with increased efficiency and reduced cost. Finally, while this is valuable work for the area of blast furnace production, more generally this work represents a triumph for the data driven approach over traditional, static, deterministic modeling techniques.

VIII. References

Akay M., Akay Y.M., Welkowitz W., "Neural networks for the diagnosis of coronary artery disease", *Proceedings of the International Joint Conference on Neural Networks*, Baltimore, MD, 1992.

Bansal K., Vadhavkar S., Gupta A., "Neural Networks Based Data Mining Applications for Medical Inventory Problems", *Journal of Agile Manufacturing*, 1998.

Bhat N. and McAvoy T.J., "Use of neural nets for dynamic modeling and control of chemical process systems", *Computational Chemical Engineering*, vol 14, pp 573-582, 1990.

Biswas A.K., *Principles of Blast Furnace Ironmaking*, SBA Publications, 1984.

Box G. and Jenkins G., *Time series analysis, forecasting and control,* Holden-Day, 1976.

Burl M.C., Fayyad U., Smythe P., Perona P., Burl M.P., "Automating the hunt for volcanoes on Venus", *Proceedings of the 1994 Conference on Computer Vision and Pattern Recognition: CVPR-94*, IEEE Computer Society Press, 1994.

Chao Y.C., Su C.H., and Huang H.P., "The adaptive autoregressive models for the system dynamics and prediction of blast furnace", Chem. Eng. Commun. 44, pp 309-330, 1986.

Fayyad U., Piatetsky-Shapiro G., Smythe P., and Uthurusamy U., *Advances in Knowledge Discovery and Data Mining*, AAAI Press/MIT Press, 1996.

Goldberg D., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

Hamilton J.D., *Time Series Analysis*, Princeton University Press, 1994.

Hirata T., Yakamura K., Morimoto S., and Takada H., "Blast furnace operation system using neural networks and knowledge-base", *6th International Iron and Steel Congress*, Nagoya, Japan, 1990.

Hush D.R. and Horne G.B., "Progress in supervised neural networks: What's new since Lippmann?", *IEEE Signal Processing Magazine*, pp 8-38, 1993.

Karhunen J., "Neural approaches to independent component analysis and source separation", *Proc. 4th European Symposium on Artificial Neural Networks (ESANN '96)*, 1996.

Lippman R.P., "An introduction to computing with neural nets", *IEEE ASSP Magazine*, April 1997.

Mallick S.K. Basu, "Automation in Blast Furnace: Scope and Recent Trends". *Tata Search*, Pp 89-96, 1998.

Pal S. K. and Mitra S., "Multilayer perceptron, fuzzy sets and classification", *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, September 1992.

Pandey B.D., et al, "G Blast Furnace at Tata Steel: A Success Story". *Tata Search*, pp 4-9, 1997.

Quinlan J.R., *C4.5: Programs for Machine Learning*, Morgan Kaufman, 1993.

Rumelhart D. and McClelland J. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, MIT Press, 1986.

Steel in the UK website, http://www.steel.org.uk, 1999.

Ullmann's Encyclopedia of Industrial Chemistry, 5th edition, vol. A 14, pp 517-540, 1985.

Ungar L.H., Powell B.A., and Kamens S.N. "Adaptive networks for fault diagnosis and process control", *Computational Chemical Engineering*, Vol. 14, pp 573-582, 1990.

Uusi-Honko, H., "Prediction of hot metal silicon in blast furnace by multivariate time series modeling", Technical report 88-90-A, Process Design Lab. Abo Akademi, Abo, Finland 1988.

Vapnik V., *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995.

Weigend A., and Gershenfeld N., eds., *Predicting the future and understanding the past*, Addison-Wesley, 1993.

Weigend A., Huberman G., and Rumelhart D., "Predicting the Future: A Connectionist Approach", *IJNS* 1 No 3, pp 193-209, 1990.

Ydstie, B. E., "Forecasting and control using adaptive connectionist networks", Comput. Chem. Engng. 14, pp 583-599, 1990.

THESIS PROCESSING SLIP

FIXED FIELD: ill. name

index biblio

► COPIES: [✓ Archives] Aero Dewey [✓ Eng] Hum

Lindgren Music Rotch Science

TITLE VARIES: ► □

NAME VARIES: ► ☑ Scott

IMPRINT: (COPYRIGHT)

► COLLATION: 57 l

► ADD: DEGREE: ► DEPT.:

SUPERVISORS:

NOTES:

cat'r: date:

page:

► S32

► DEPT: E.E.

► YEAR: 1999 ► DEGREE: M.Eng.

► NAME: BANKS, Bradley S.