CHAPTER 2

LITERATURE SURVEY

2.1 Basic Theory of Neural Network

Neural network is an information-processing system that has certain performance characteristic in common with biological network [1]. It is the high interconnected neural computing elements that have the ability to respond to input and to learn to adapt to the environment. Neural network simulates the working of human brain. The basic computing elements in biological network are neuron. A neuron is a small cell that receives electrochemical stimuli (input) from many sources and responds by generating electrochemical impulse and transmits to other cells on action of human.

Neural networks have been shown the effectiveness as a computational processor for various tasks such as pattern recognition (speech and visual image recognition), classification, data compression, modeling, forecasting, adaptive control. They have a number of desirable properties which are not found in conventional computation system including the dealing with noisy or incomplete input patterns, high parallel computation rates and adaptive learning.

2.1.1 Structure of Neural Network

Neural network consists of many processing units, called neuron or node. Each neuron is arranged in layer. Input layer is the layer that neurons receive inputs from external and does not recompute their outputs. It is just only passing the

input data to the next layer. Output layer is the layer that presents the output of whole network by computing their output from input data. The layer between input layer and output layer is called hidden layer. It is called hidden layer because the neurons in this layer receive only internal inputs (inputs from other processing units) and compute internal outputs (outputs to other processing units). This layer is hidden from outside of network. This layer is necessary for a network to compute difficult, complex, non-linear function. Each node in each layer is connected together by weight. The number of neuron in input layer is equal to the input parameter of problem and the number of neuron in output layer is equal to the output parameters. But for hidden layer, the number of neuron is dependent on characteristic or complex of problem on hand. The arrangement of neurons into layers and connection pattern between layers is called the architecture of network.

2.1.2 Basic Concept of Neural Network Computing

In Figure 2.1, a single neuron network is illustrated with three inputs and one output. Each input neuron *i* (i=1, 2, 3,…,i) has an external input X_i and weight $W_{i,j}$ which is part of connection of neuron to neuron.

The neuron has a function for transferring or producing an output

$$y = f(\mathbf{I})$$

$$\mathbf{I} = x_1w_1 + x_2w_2 + x_3w_3 = \Sigma x_iw_i$$

$$y = f(\Sigma x_iw_i)$$

where **I** is cumulative input to the neuron.

$f(\ )$ is an activation function which computes the output

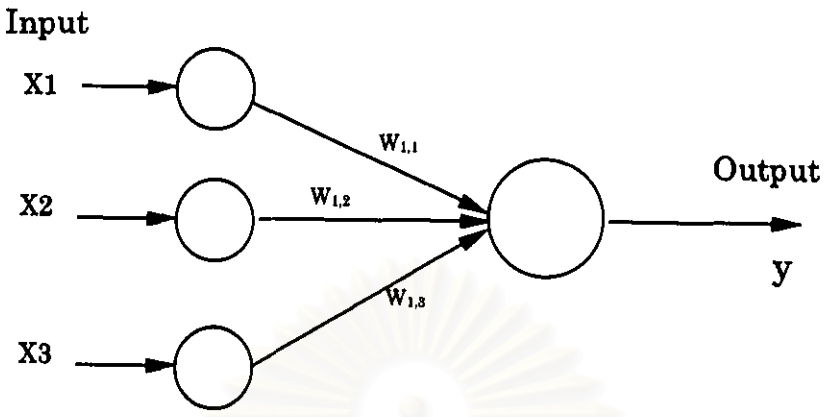

Figure 2.1 A Simple Neural Network

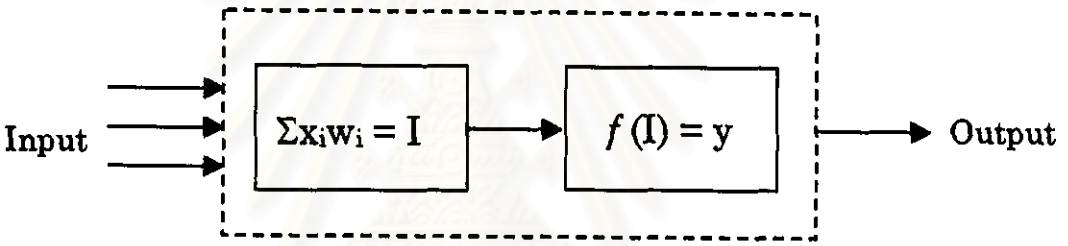

Figure 2.2 Schematic Diagram of a Single Neuron

2.2 Basic Concept of Multilayer Perceptron Network with Backpropagation

The first neural network was designed by Warren McCulloch and Walter Pitts in 1943 **[2]**. Many neural networks had been designed to solve different complex problems such as Kohonen network, Recurrent network, Adaline network, and Hopfield network. One of the most important neural networks is the multilayer perceptron network with backpropagation. It is a highly flexible modeling tool.

A general multilayer perceptron network is presented in Figure 2.3. It is a feedforward and fully connected network. It consists of an input layer, one or more hidden layer and an output layer. The iteration of network is started with presenting of real value of input x_i to input layer and passed to the first hidden layer units through weights w_{ji}. Unit j in hidden layer computes its output by using x_i and w_{ji} and passes output to the units in the next layer. In the next layer, it is like in the previous layer it receives input from previous layer and compute the output then pass to the next layer. This process is repeated until the output units. Multilayer perceptron with backpropagation training algorithm has gained popularity because it can perform arbitrary mapping. Arbitrary mappings are possible if a sufficient number of hidden units are provided and if the network can be trained and the weights that perform the desired mapping can be found. The problem is how to find the set of this weights. The backpropagation learning method specify how to adjust the weights in each layer. It is an optimization procedure base on gradient descent that adjusts weights to reduce the error of system. During the learning of network, each training data set is propagated forward layer by layer until an output of network is obtained. The output of network is then compared to a target output and an error is determined. This error is used as input to feedback connection which adjustments of weight are made layer by layer in a backward direction. The name backpropagation come from this backward direction. When the new weight is obtained after adjustment, they are used for next iteration. And the backward direction is used for adjustment. The process is continued in the training phase until the system error convergence to a minimum or until some limit is reached.

To make understanding of calculation of multilayer perceptron network, we determine the following notation for the network parameters. From Figure 2.3, weight connection between input layer unit i and hidden layer unit j are denoted by v_{ji} i = 1,2,...,i j = 1,2,...j and weight connections between hidden layer unit j and output unit k are determined as w_{kj} , k = 1, 2, ...,k. The input is denoted as x_i and output of hidden units j is y_j while output of unit k in output layer is z_k , target output is t_k.

We define the following terms

$$H_j = \Sigma v_{ji} x_i$$

$$I_k = \Sigma w_{kj} y_j$$

$$y_j = f(H_j)$$

$$z_k = f(I_k)$$

H_j = the net input to hidden layer unit j

I_k = the net input to unit k of output layer

y_j, z_k is output computed by unit j of the hidden layer and unit k of output layer respectively

$f(\ )$ = activation function which is as bounded, differentiable function

Thus $$z_k = f(I_k) = f(\Sigma w_{kj} y_i) = f(\Sigma w_{kj} f(H_j))$$

$$= f(\Sigma w_{kj} f(\Sigma v_{ji} x_i))$$

After we get output, we need the algorithm that reduce the error system through an adjustment of weights. We define the mean system error E_t as the average of the output errors over all training set error E_p , p = number of training set data.

$$E_t = 1/p^* \Sigma E_p$$

$$E_p = 1/2\ \Sigma(t_k - z_k)^2$$

The system error will be reduced if the error of each training set, E_p, is reduced. The parameter that can be adjusted in network is only the weights in the network. Thus a weight correction procedure must be the procedure that adjusts the weights in proportion to a reduction in the error relative to changes in the weights. We will change the weights on each successive set presentation such that the set errors are iteratively reduce from previous values. Thus at step s+1 of the training process, the weight adjustment should be proportional to the derivation of the error of E_p on iteration s

$$\Delta w(s+1) = -\eta \partial E_p / \partial w(s)$$

$$w_{kj}(s+1) = w_{kj}(s) + (-\eta \partial E_p / \partial w_{kj}(s))$$

η = a constant learning coefficient (learning rate)

$$E = 1/2\ \Sigma(t_k - z_k)^2 \qquad z_k = f(I_k),\ I_k = \Sigma w_{kj} y_j \qquad \textbf{(2.1)}$$

From chain rule

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial I_k}\frac{\partial Ik}{\partial w_{kj}} = \frac{\partial E}{\partial I_k}\left(\frac{\partial \Sigma y_j w_{kj}}{\partial w_{kj}}\right)$$

$$\frac{\partial \Sigma y_j w_{kj}}{\partial w_{kj}} = y_j$$

We use chain rule again

$$\frac{\partial E}{\partial I_k} = \frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial I_k}$$

From equation (**2.1**) $$\frac{\partial E}{\partial z_k} = -(t_k - z_k)$$

$$\frac{\partial z_k}{\partial I_k} = f'(I_k)$$

We define $$\delta_k = (t_k - z_k) f'(I_k)$$

Thus, for output layer $$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} = \eta \delta_k y_j$$

For hidden layer $$\Delta v_{ji} = -\eta \frac{\partial E}{\partial v_{ji}} = -\eta \frac{\partial E}{\partial H_j} \frac{\partial H_j}{\partial v_{ji}}$$

$$\frac{\partial H_i}{\partial v_{ji}} = \Sigma \frac{\partial}{\partial v_{ji}} (v_{ji} x_i) = x_i$$

$$\frac{\partial E}{\partial H_j} = \frac{\partial E}{\partial H_j} \frac{\partial y_j}{\partial H_j} = \frac{\partial E}{\partial y_j} f'(H_j)$$

$$\frac{\partial E}{\partial y_j} = \frac{1}{2} \Sigma \frac{\partial (t_k - f(\Sigma w_{kj} y_j))^2}{\partial y_j}$$

$$= -\Sigma (t_k - z_k) f'(I_k) w_{kj}$$

We can represent as $$\Delta v_{ji} = \eta \delta_j x_i = \eta x_i f'(H_j) \Sigma \delta_k w_{kj}$$

Thus for weight adjustment in output and hidden layer we use equation (**2.2**) and (**2.3**)

$$w_{kj}^{new} = w_{kj}^{old} + \Delta w_{kj} = w_{kj}^{old} + \eta y_j (t_k - z_k) f'(I_k) \qquad \text{Equation (2.2)}$$

$$v_{ji}^{new} = v_{ji}^{old} + \Delta v_{ji} = v_{ji}^{old} + \eta x_i f'(H_j) \Sigma \delta_k w_{kj} \qquad \text{Equation (2.3)}$$

The backpropagation training process requires the activation function that is bounded, differentiable functions and its differentiate.function can be written in form of its function. One of the most commonly used function is sigmoid function or logistic function.

$$f(x) = 1/(1+e^{-x})$$

$$f'(x) = f(x)(1-f(x))$$

Another commonly used function is the hyperbolic tangent function. This function has a shape similar to the sigmoid function. It can be written as

$$f(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x})$$

$$f'(x) = (1-(f(x))^{2})$$

For improving the rate of convergence, adding some momentum to the weight adjustment expression is done. This is done by adding a fraction of the previous weight change to the current weight change. This can prevent extremely change in some adjustments. We can represent the new adjustment equation as:

$$\Delta w_j(t+1) = -\eta \frac{\partial E}{\partial w_j}(t) + \alpha \Delta w_j(t)$$

where α is the momentum coefficient. The value of α should be positive and less than 1. Typical value is in the range [0.1-0.9]. But some problem momentum = 0 can learn better.

2.3 Multilayer Perceptron Network with Backpropagation Algorithm

1. Initialize all weights W to small random values within the range [-λ, λ]. Normally all weights in range [-0.1, 0.1] are effective values.

2. Randomly select a pair of training set {x, t} and compute in a feedforward direction the output values for each unit j of each layer, thus

$$O_j = f(\Sigma O_i w_{ji})$$

3. Use the values O_j computed by the final layer units and the corresponding target values t_j to compute the delta quantities

$$\delta_k = (t_k - O_k) f'(I_k)$$

4. Compute the deltas for each unit in preceding layers by using

$$\delta_j = f'(H_j) \Sigma \delta_k w_{kj}$$

5. Update all weights w_{ji} using

$$w^{new} = w^{old} + \Delta w$$

6. Return to step 2 and repeat for each set until the acceptable error is reached.

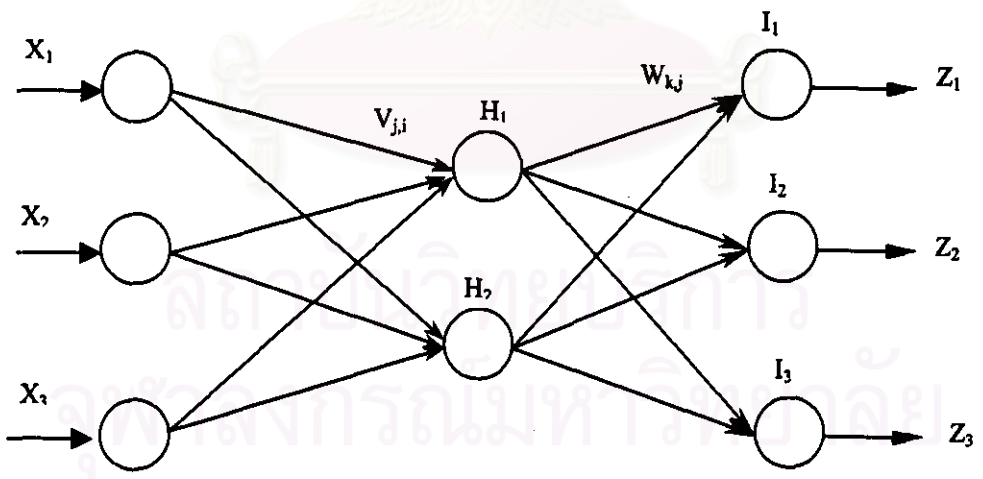

Figure 2.3 A general Multilayer Neural Network

2.4 Application of Neural Network in the Iron and Steel making process control

2.4.1 Blast Furnace Analysis with Neural Networks

In the operation of blast furnace the radial temperature profile in the upper part of furnace is an important factor which controls the quality of process. This value is the important parameter to describe the condition of furnace. The optimization of this temperature profile may lead to saving of the production cost. To achieve this optimization, quantitative relations between furnace parameter are needed but those relationships are unknown. A process model can be provided by using neural network [3]. To build the model the data for modeling the blast furnace consist of the charging program, the distribution of the grain size of coke and sinter, several parameter of the material strength, material transportation time. These data were used as the input of neural networks. The outputs of the network are the value of modeling temperature at 8 positions of the measuring system. Several network configurations have been trained and tested. The best configuration (the minimum error) was chosen as the model for this process. The neural network is able to approximate the temperature profile with good precision. By using this model the optimization of the blast furnace process can be largely improved. The results of this control system are shown in Figure 2.4.

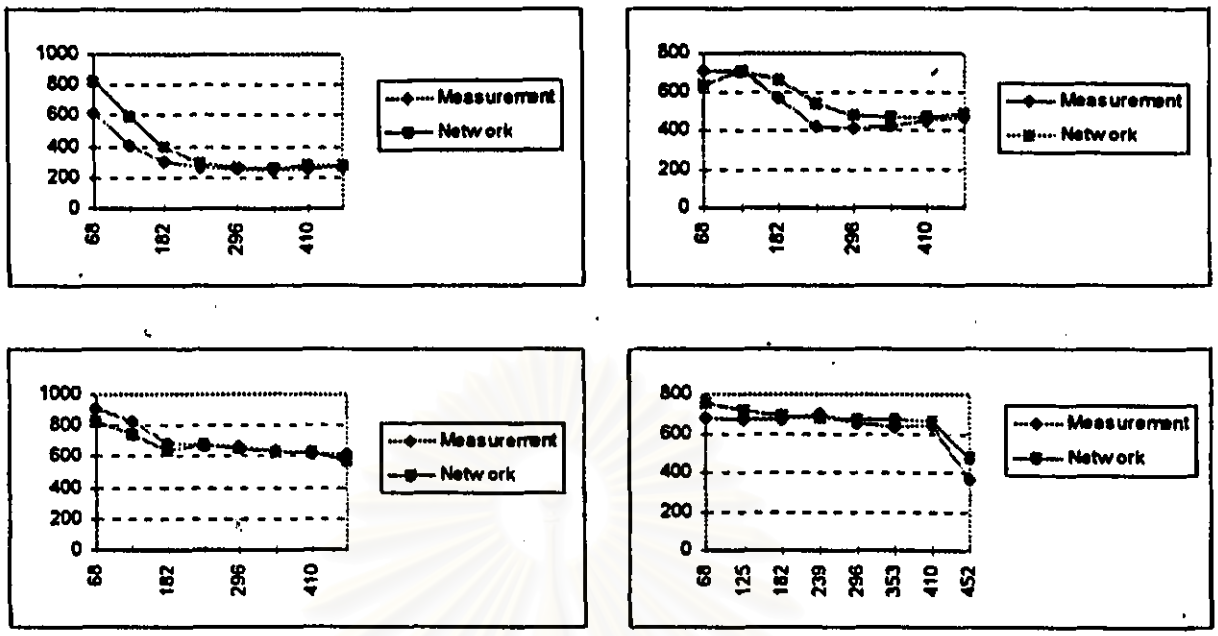

Figure 2.4 The result of temperature profile with neural network control

2.4.2 Neural Model of Blast Furnace Burden Distribution

The burden distribution of blast furnace is a factor for achieving a gas distribution in the way that it ensures the efficiency and a smooth operation of the blast furnace. The burden distribution cannot be measured. However temperature measurements can be used to estimate the burden distribution **[4]**. The local layer thickness of the charged burden causes the changing of temperature. The neural network is presented for modeling the blast furnace burden distribution by using the data of temperature measurement. The data for modeling the burden distribution consists of the index which are derived from temperature measurement (equation 2.4), the time elapsed between dumping materials, binary variables indicating the material (1 for ore and 0 for coke), particle size (1 for fine and 0 for coarse) and temperature.

$$\beta_i = \frac{T_{gi}(t_d) - T_{gi}(t_d + 2)}{T_{gi}(t_d) - T_s} \qquad i = 1, 2, 3, ..., K \qquad \text{(Eq 2.4)}$$

where T_s is charged burden, T_g is temperature, t_d is the time of the dump, and K is the position of measuring temperature.

Several feedforward networks have been trained and tested. The best feed forward network with seven input, one output and five hidden nodes was found as the model of burden distribution accurately and is shown in Table 2.1.

Network	Number of weights	train error	Test error
(7,1)	8	0.0763	0.0767
(7,2,1)	19	0.0659	0.0703
(7,3,1)	28	0.0602	0.0714
(7,4,1)	37	0.0564	0.0714
(7,5,1)	46	0.0555	0.0636
(7,6,1)	55	0.0513	0.0706
(7,7,1)	64	0.0501	0.0691
(7,8,1)	73	0.0481	0.0653
(7,9,1)	82	0.0451	0.0688

Table 2.1 The training and testing error for different architecture of 7 inputs network

The predicted value of β is compared with the true value in the Figure 2.5.

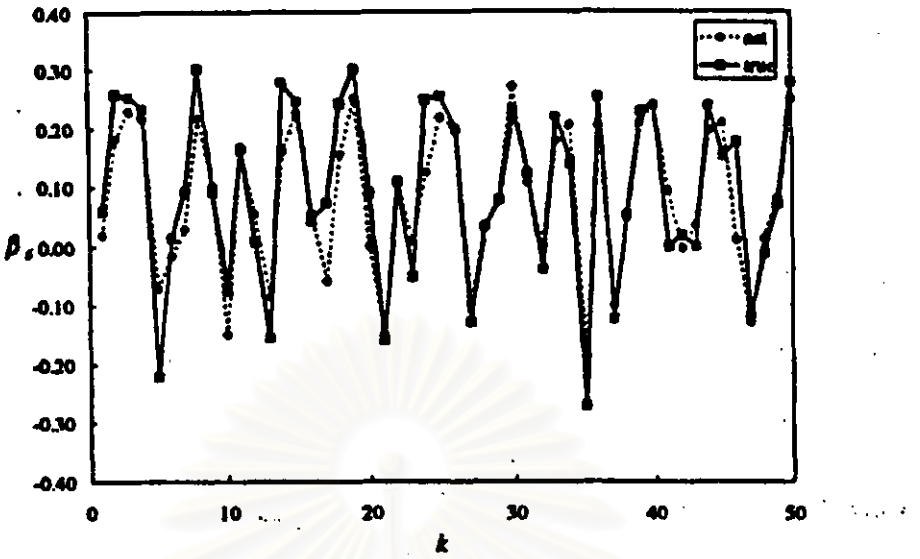

Figure 2.5 The true values of β and predicted values by the [7,5,1] network

2.4.3 Optimization of a BOF's Lining Resistance by Application of Artificial Neural Network (ANN)

This work had been done at a Chinese steel plant. The furnace life of the basic oxygen furnace in this steel plant is about 1000 heats of steel and the production rate of 30 heats/day [5]. Consequently the furnace must be stopped for changing of the lining material every month. The cost for lining is very expensive. In order to improve the refractory of furnace, the relationships which affect the furnace life should be known. These relationships are complex therefore neural network was used to model the furnace life. A data set of this work contains 18 variables. These variables are:

1. X_1 : supplementary filling amount (kg/t steel)
2. X_2 : percentage of first hit (%)

3. X_3 : blowing time
4. X_4 : waiting time
5. X_5 : melting time
6. X_6 : Si content in molten iron
7. X_7 : Mn content in molten iron
8. X_8 : P content in molten iron
9. X_9 : S content in molten iron
10. X_{10} : temperature of molten iron
11. X_{11} : temperature of final point
12. X_{12} : CaO amount (kg/t steel)
13. X_{13} : dolomite amount (kg/t steel)
14. X_{14} : molten iron ratio (%)
15. X_{15} : total iron content in the residue
16. X_{16} : MgO content in the residue
17. X_{17} : alkaline degree (CaO/SiO_2)
18. X_{18} : production rate (%)

First the DDL (Distance Discriminate Line) method is adopted to analysis 18 variables. It was found that X_1, X_3, X_5 and X_{18} were major factors that determine the furnace life.

The data set were divided into 2 classes

Class 1. Good samples: the furnace life is longer than 1000 heats

Class 2. Bad samples: the furnace life is shorter than 1000 heats

The total amounts of data sets were 33 sets. 16 data sets were classified into Class 1 and 17 data sets were classified into Class 2. Four variables (X_1, X_3, X_5 and X_{18}) were

taken as the input, the expected output of these two classes were defined as 0.1 (class 1) and 0.9 (class 2). The hidden layer contains 8 neurons. 25 data sets were used for learning and the other 8 data sets for testing. The results of testing are shown in Table 2.2.

X_1	X_3	X_5	X_{18}	Predicted result by ANN	Predicted class	Actual class
0.1840	16.2	35.8	44.8	0.011819	1	1
0.2486	17.7	37.2	37.9	0.108319	1	1
0.1679	17.1	34.6	37.3	0.003558	1	1
0.2222	18.4	37.0	42.9	0.086422	1	1
0.2445	18.2	86.6	37.9	0.005274	1	1
0.0816	20.9	41.2	52.6	0.904718	2	2
0.1465	19.1	38.6	28.1	0.875118	2	2
0.1008	18.2	37.0	33.9	0.992812	2	2

Table 2.2 The predicted results by ANN

From the result of testing, it can be seen that the prediction was correct according to the actual class. This result can be applied to adjust the main factors which effect the wear of the lining of BOF (X_1, X_3, X_5, X_{18}).

2.4.4 Adaptive Neural Net (ANN) Models for Desulphurization of Hot Metal and Steel

In steelmaking process desulphurization is a step in secondary metallurgy which should be achieved because sulfur can form inclusions with other elements which may affect the mechanical properties of steel negatively.

Desulphurization can be done in several parts in iron and steel making such as in converter, in torpedo car or in vacuum degassing unit. There are many factors that affect desulphurization. In this work neural network was adapted to predict the sulfur content of hot metal at the end of calcium carbide powder injection into a 400 t torpedo ladle. Furthermore the sulfur content of steel at the end of blowing in a 300 t converter was predicted **[6]**. The prediction of the sulfur content of hot metal used hot metal weight, treatment time, initial sulfur content, and gas flow rate and powder injection rate as the input and the final sulfur content as the output of neural network. The results of prediction are shown in Table 2.3. For the prediction of sulfur content in oxygen steelmaking, eight variables were used:

1. metal weight
2. total amount of oxygen blown
3. amount of iron ore added
4. temperature
5. contents of carbon
6. content of manganese
7. content of phosphorus
8. content of sulfur

These variables were used as inputs of neural network for predicting the sulfur content of steel whereas the sulfur content of liquid steel at the tapping was an output variable. The neural networks were trained with 50 data sets and tested with 50 data sets. The result of testing is shown in Figure 2.6. From these two results it is obvious that neural network can be applied to model the desulphurization of hot metal and steel.

No	Time (s)	Weight (ton)	Initial S Content %	Gas flow rate m^3/min.	Powder flow rate Kg/min.	Final S content in %			
						Actual	Predicted		
							(16)* [5,8,1]#	(24)* [5,8,1]#	(40)* [5,9,1]#
1	1404	280	0.026	0.6	67.0	0.003	0.1436	0.0059	0.0030
2	1421	309	0.024	0.6	70.0	0.003	0.0150	0.0031	0.0030
3	473	303	0.016	0.6	62.0	0.005	0.0074	0.0033	0.0084
4	1161	362	0.030	0.6	73.0	0.005	0.0157	0.0030	0.0030
5	1085	336	0.028	0.6	63.0	0.008	0.0143	0.0090	0.0050
6	559	308	0.021	0.6	63.0	0.009	0.0090	0.0036	0.0149
7	483	304	0.018	0.6	58.0	0.011	0.0087	0.0043	0.0030
8	501	317	0.020	0.6	63.0	0.011	0.0091	0.0038	0.0096
9	779	295	0.030	0.6	68.0	0.012	0.0159	0.0161	0.0163
10	796	344	0.033	0.6	60.0	0.015	0.0160	0.0161	0.0163
11	455	271	0.022	0.6	68.0	0.017	0.0151	0.0042	0.0166

Table 2.3 The results of predicted final sulfur in torpedo ladle with different architecture network

* number of training patterns # architecture of neural network [5,8,1] = 5 input neurons, 8 hidden neurons, 1 output neurons

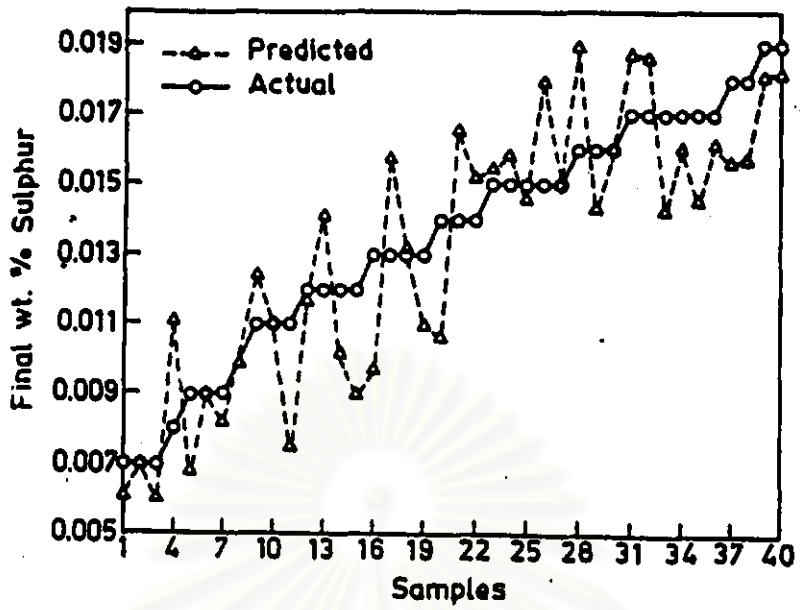

Figure 2.6 The result of prediction of final sulfur content in blowing converter

2.4.5 Modeling of Metal-slag Equilibrium in Ironmaking Process Using Neural Network

The existing fundamental models for the describing metal-slag equilibrium, for example, Temkin's rule, Schuhmann's approach, Masson's polymer theory, the model of Flood and Grjotheim, use the complicated mathematical equations. They are not easy to be used in practice. This example will show the new alternative for modeling metal-slag equilibrium by using neural network [7]. This example illustrates the application of neural network modeling for ironmaking process. Two equilibrium were considered. The first is manganese distribution between pig iron and slag. This investigation used the data from an article of Oelsen and Schubert. Two input variables were used, %CaO/%SiO_2 and[Si]. The output was [%Mn]/(%Mn) ratio. The system also used eight hidden nodes. The neural network was trained with

48 data sets and tested with 7 data sets. The error after convergence of testing is 4%. Figure 2.7 plots the predicted [%Mn]/(%Mn) ratio over the experimental [%Mn]/ (%Mn) ratio. This figure shows that the neural network can predict [%Mn]/ (%Mn) ratio correctly.

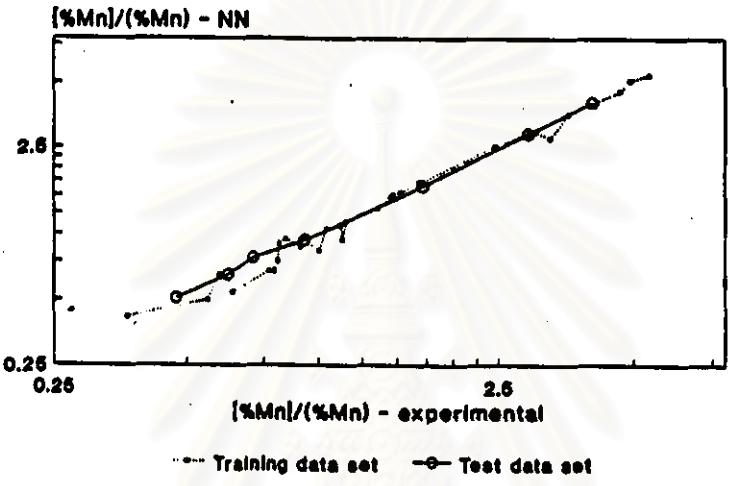

Figure 2.7 The predicted [%Mn]/(%Mn) by neural network and by experiment

The second application was the prediction of the sulfur distribution between pig iron and slag. Taylor and Stobo investigated the sulfur distribution between slag and pig iron at 1500°C in an aluminium crucible. The data from this investigation have been used in this work. The neural network included four input nodes, five hidden nodes and one output node. The input data were (%CaO)/(%SiO_2), %Al_2O_3, %FeO and %S. The output is (S)/[a_s]. The error testing data is 6.6%. This shows that neural network can predict proper the distribution of S between pig iron and slag. The results of neural network are shown in Figure 2.8.

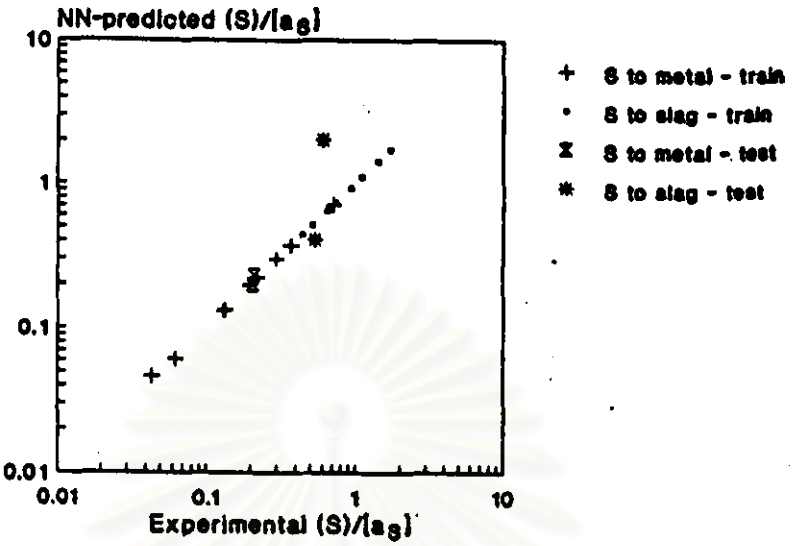

Figure 2.8 A comparison between NN-predicted (S)/[a_s] and experimental (S)/[a_s]

2.4.6 Neural Network System for Breakout Prediction in Continuous Casting Process

In this example the neural network is used for the prediction of breakout in continuous caster mold [8]. In the continuous casting process the breakout of caster mold is the gravest factor that obstructs the process because it may cause the long shutdown of continuous caster. The breakout is a phenomenon where the solidifying shell of molten steel in contact with the mold wall ruptures and causes the molten steel spill out when the rupture reaches the bottom of the mold with the progress of the casting operation. If this shell rupture can be verified by some means, reducing the casting speed can prevent the breakout. The occurring of breakout can be verified by measuring temperature at mold wall because the molten steel directly

contacts the mold wall where the shell rupture is located. Therefore the measured temperature at the mold wall is the input of the neural network and the output is a breakout alarm when the output value exceeds a predetermined threshold value. The result of this neural network is shown in Figure 2.9. The conventional method had 0 miss item and 23 overdetection items, whereas the neural network system had a 0 miss item and 0 overdetection item. This network brought an accuracy of almost 100% and proved the system capability of predicting breakout is better than the conventional method.

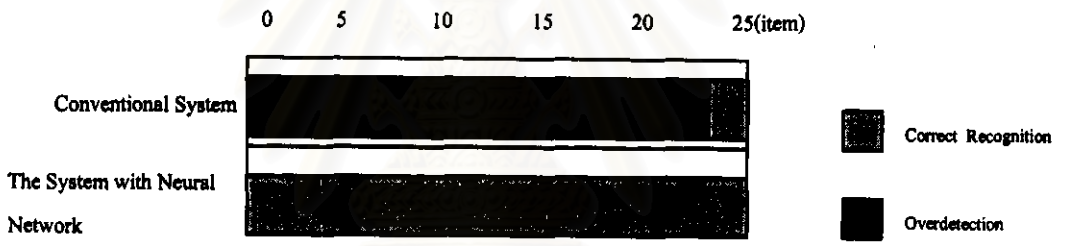

Figure 2.9 Result of simulation test

2.4.7 Use of Artificial Neural Networks on Optimization of Mechanical Properties of Batch Annealed Thin Steel Strip

The optimization process can be constructed by an accurate model with the absolute understanding of the process. Some of the mechanical properties of the strip are mainly adjusted with the batch anneal process. During the simulation of the process the development of a multilayer perceptron model for the optimization of mechanical properties of batch annealed thin steel strips was developed [9]. This work

got the data from the process of Rautaruukki's Haemeenlinna and Raahe works, which were collected continuously for a period of approximately 6 months. The input data were uniaxial tension test and the concentration of most important alloying elements (C, Si, Mn, P, Nb, and Ti). The output of the model was yield strength. The multilayer perceptron used back propagation rule to find an optimum mapping between input and output. This multilayer perceptron contained only one hidden layer, which is sufficient to represent a mapping between input and output. The optimization algorithms consisted of two multilayer perceptrons (MLP). One for calculating the yield strength of the end product (prediction MLP) and the other for giving an estimation for the credibility of the prediction yield strength (evaluation MLP). The interaction of alloying elements is non-linear. The behavior of the two MLPs were tested with data where four of the alloying elements were at the average values of a certain steel grade and two of them (niobium and titanium) were changed between their minimum and maximum. The results of this test are shown in Figures 2.10 and 2.11.

Figure 2.10 shows that the predicted yield strength increases with the addition of Nb and Ti. The magnitude of the increase is in agreement with experimental knowledge. Thus it seems that the prediction MLP has also approximated the effect of niobium and titanium correctly in the other regions.

Figure 2.11 shows the credibility estimate predicted yield strength by evaluation MLP. Typically the contents of niobium and titanium of this investigated steel grade are at the top left corner of Figure 2.11 which shows poor credibility. Because this two MLPs were tested with a constant value of the other four alloying elements which should affect the yield strength of steel.

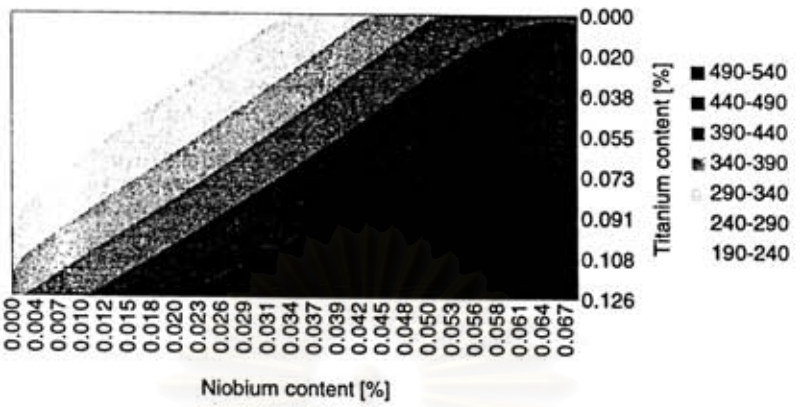

Figure 2.10 The predicted yield strength with addition of Nb and Ti

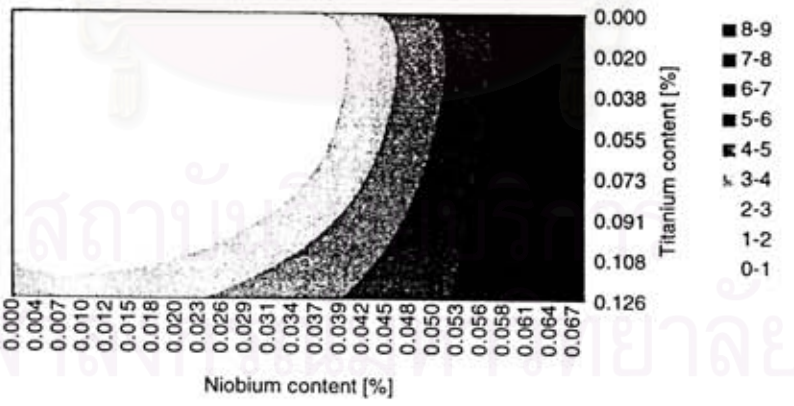

Figure 2.11 The credibility estimate predicted yield strength by evaluation MLP

2.4.8 Improvement of Cold Mill Precalculation Accuracy Using a Corrective Neural Network

In the cold rolling process the stands of rolls are used to flatten a steel strip to a desired thickness. The steel strip is pressed while passing the stands of rolls. One of the most important variables, which affect on the flattening of a steel strip are roll gap and rolling force. At Pohang Iron and Steel Company (POSCO) in Pohang/Korea these variables were determined by precalculation phase of cold mill process control. Before using the neural network POSCO used mathematical models to determine rolling force and rolling gap in precalculation phase. The exact values of some parameters such as, friction coefficients, deformation resistance of coil, to calculate rolling force and rolling gap was not known. Because these parameters could not be measured during processing of coil. Another problem in mathematical model is that it did not use some important parameters which influence the rolling force, e.g. the coiling temperature behind the last stand of the hot mill, the chemical composition of the coil, the aggregated amount of processed strip at each stand and the roll type. Therefore POSCO developed the precalculation phase by using corrective neural network **[10].** This control system combined the mathematical model and neural network as shown in Figure 2.12.

The output of the corrective neural network is the corrective coefficient Z_p, the ratio of the actual measured rolling force (P_{act}) to the predicted rolling force (P_{cal}) by mathematics model. This corrective coefficient is multiplied to the rolling force calculated by mathematical model to result in the combined rolling force (P_{comb}). A

corrective neural network is built for each of the five milling stands. The result of this development is shown in Table 2.4. The error (E) is defined as the difference of rolling forces.

$$E_{mat} = |P_{act} - P_{cal}|$$

$$E_{comb} = |P_{act} - P_{comb}|$$

From the result in Table 2.4 the applied neural network to precalculation phase of POSCO can reduce the error 55.4% on average and up to 72% on stand No 2.

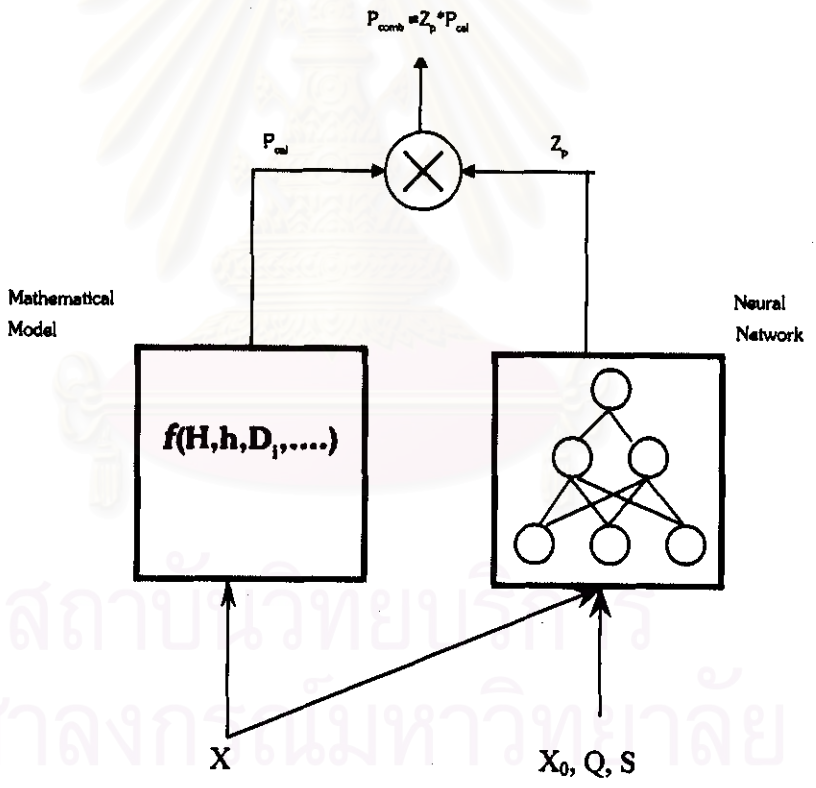

Figure 2.12 Combined model of mathematical and neural network

X = { roll diameter (D_i), forward tension (T_f), backward tension (T_b), initial thickness (H), target thickness(h), coil width(W) }

X_0 = { aggegrated amount of coil processed, roll type }

Q = { chemical composition (C, Mn, Si) }

S = { average coiling temperature at hot rolling mill }

Stand Number	E_{mat}(ton)			E_{comb}(ton)			average error reduction (%)
	avg.	sdv	max	avg.	sdv	max	
1	118.82	87.55	297.00	41.48	42.27	213.83	65.1
2	98.09	43.22	295.00	27.17	22.09	164.00	72.3
3	116.83	71.81	290.00	36.85	34.66	169.71	68.5
4	35.46	29.33	137.00	30.37	28...76	137.02	14.8
5	41.08	29.34	163.00	17.97	19.15	142.00	56.2
Average	82.09	52.25	236.40	30.77	29.38	165.31	55.4

Table 2.4 Comparison of the rolling force prediction error both the mathematics and combined model

2.4.9 Prediction of martensite start temperature using artificial neural network

This investigation uses the artificial neural network to predict the martensite start temperature **[11]**. Martensite is one important microstructure that affects the hardness of steel. Martensite is a desired microstructure in some heat treatment processes. Martensite start temperature (M_s), i.e. the highest temperature where the martensite phase occurs. It can be predicted from the chemical composition of steel because the austenite-martensite transformation is a non-diffusional transformation therefore the characteristic transformation temperature does not depend on the prior thermal history during cooling or on the austenite grain size. The new modeling technique, which is presented to predict martensite start temperature, was Neural

Network. Neural network is very stable and capable of handling linear and non-linear dependencies as well as higher order parameter interactions **[12].** The chemical composition of steel was the input of neural network and the output was the martensite start temperature. This work used 164 vanadium steel grades and the measured M_s from the atlas of continuous cooling transformation diagrams for vanadium steel. From these 164 steel grades, 144 steel grades were used for training and the other 20 steel grades for testing. The 12 elements which have significant effect on M_s were used as in put: C, Si, Mn, P, S, Cr, Mo, Ni, Al, Cu, N and V. The neural networks were trained with many architectures. The best results were received by the architecture 6 nodes of hidden layer. The results of this architecture are shown in Figure 2.13. A good correlation was obtained in a range of 180-480°C. Figure 2.14 show the results of predicted M_s by linear Andrews model. When comparison the results between the predicted M_s by neural network and by linear Andrews model, the results from neural network is better than form linear Andrews model. Therefore Neural Network can be applied to predict the martensite start temperature correctly. Moreover neural network is capable of handling different dependencies in different areas of the total composition **[11].** Four base alloys have been selected which cover the domain of input and output for looking at the effects of the alloying elements on M_s. The results of the effect of alloying elements on M_s is illustrated in Figure 2.15. In the Figure 2.15 it is found that M_s is a function of overall composition. The nickel and manganese dependencies are linear whereas the carbon and molybdenum dependencies are non-linear. The chromium and vanadium concentration does not affect the M_s.

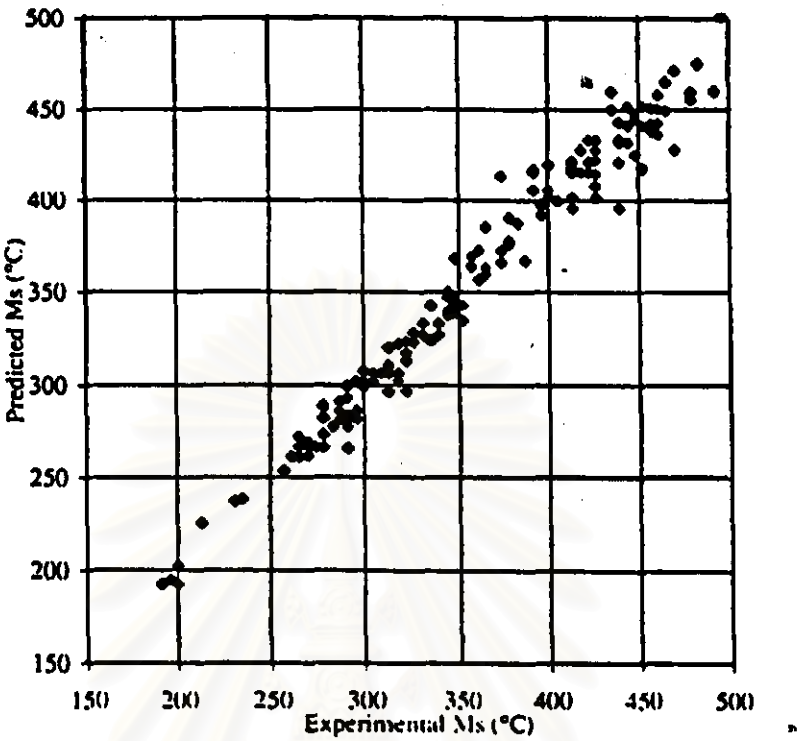

Figure 2.13 The comparison between the predicted M_s and the measured M_s

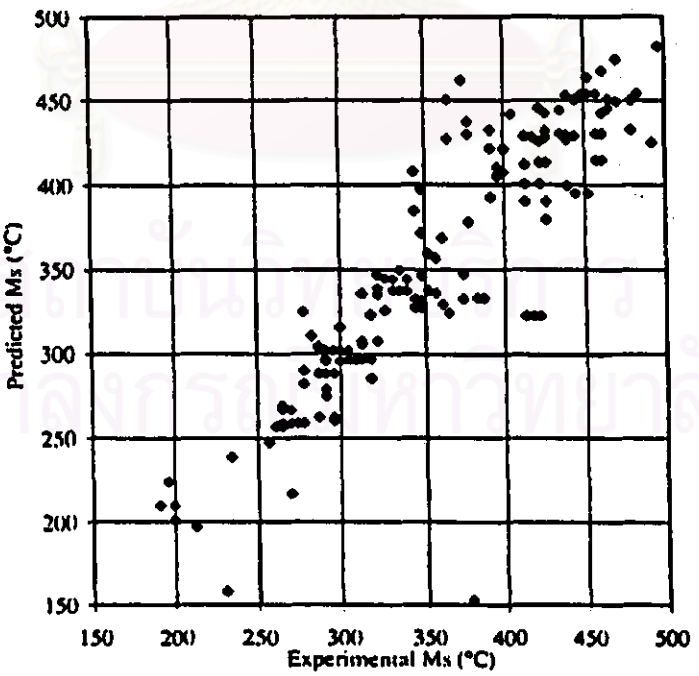

Figure 2.14 The predicted M_s by Line Andrews Model

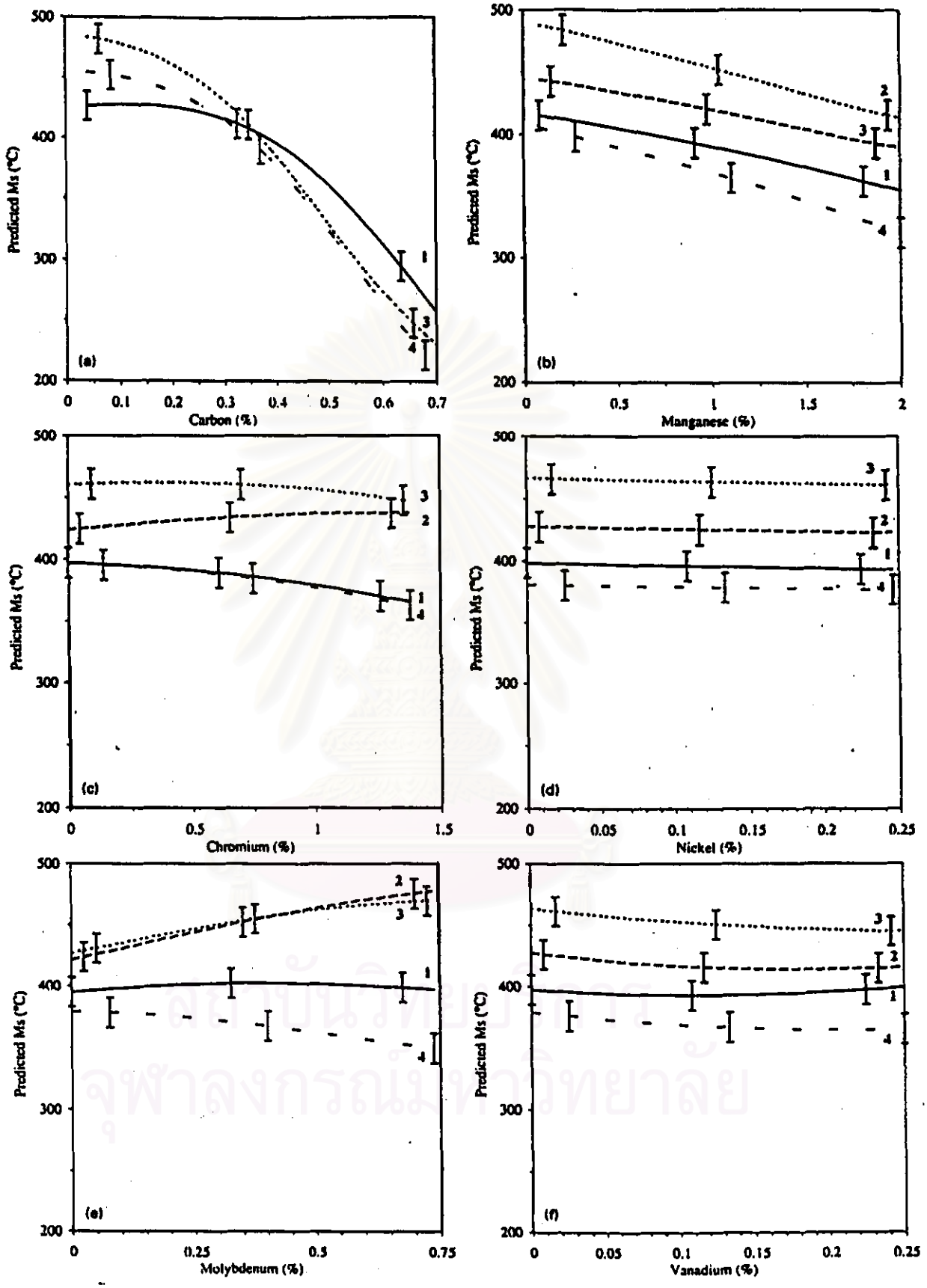

Figure 2.15 The effect of alloying elements on the M_s