WINNER TAKE ALL EXPERTS NETWORK

FOR SENSOR VALIDATION

By

WEI FENG

Bachelor of Science

Xi'an JiaoTong University

Xi'an, P. R. China

1996

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
In partial fulfillment of
The requirements for
the Degree of
MASTER OF SCIENCE
May, 2000

WINNER TAKE ALL EXPERTS NETWORK

FOR SENSOR VALIDATION

Thesis Approved:

[illegible]

Thesis Adviser

[illegible]

[illegible]

Wayne B. Powell

Dean of the Graduate College

PREFACE

The validation of data from sensors has become an important part in the operation and control of modern industrial equipment. To validate a signal, the sensor must be shown to consistently provide the correct measurements, and the analysis of the validation hardware or software should trigger an alarm when the sensor signal deviates appreciably from the correct value. Neural network based models can be used to estimate critical sensor values when neighboring sensor measurements are used as inputs. The discrepancy between the measured and predicted sensor value may then be used as an indication of sensor health.

Traditional neural network such as Multi-layer Perceptron (MLP) network and Radial Basis Functions (RBF) network have been proved successful as universal function approximators. But the training algorithm is typically too slow for solving real world problems. In addition, when the problem becomes complicated, most of the systems even could not converge to a local minimum in a reasonable time due to the limitation of the hardware.

The proposed Winner Take All Experts (WTAE) network is based on 'divide and conquer'. It employs growing fuzzy clustering methods to divide a complicated problem into a series of simple sub-problems and assign an expert to each of them. It also allocates every new case to one of the experts, and, if the output is incorrect, the weight

adaptation is localized to the local expert. As a result, it is a fast learning algorithm without knowing a priori information.

After the sensor approximation, the outputs from the estimator and the real sensor readings are compared both in time domain and in frequency domain. Three fault indicators are used to detect the sensor failure. In the decision stage, the intersection of three fuzzy sets accomplishes a decision level fusion, which indicates the confidence level of the sensor health.

Two data sets, the Spectra Quest (SQ) Machinery Fault Simulator data set and the Westland vibration data set were used in simulation experiments to demonstrate the performance of the WTAE network. Comparisons of tracking performance among the proposed network and MLP, RBF network were performed. The WTAE was found competitive with or even superior to the others. Comparisons for decision making processes between WTAE network and traditional time domain indicators were also performed. The WTAE achieved 100% correct detection for both testing data set without knowing a priori information. Using the same testing sets, the traditional indicators only detected less than 87.5% failure states depending on knowledge of characteristics of both the sensors and the environments.

ACKNOWLEDGEMENTS

I would like to acknowledge the continued support, guidance, and encouragement from my thesis adviser, Dr. Gary G. Yen. He provided deep insight and technical advice on all the issue of this research. I am grateful to the members of my committee members for taking out their time and for providing constructive feedback for my research. Thanks are also extended to Dr. Martin Hagan for his instruction and help in the Neural Network course I took two years ago.

I wish to express my strong appreciation to the Intelligent System and Control Laboratory at Oklahoma State University for supporting resources. Furthermore, I would also like to greatly thank Liang-Wei Ho, Phayung Meesad, and other people in the Intelligent System and Control Laboratory for their help and discussions.

Finally, I would like to thank my wife Wei-Ching, for her love, patience and endurance that made this journey a pleasant experience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Motivation and importance of the research

At 12:36 a.m. EDT, July 20,1999, NASA aborted the launch of the space shuttle Columbia just about 7 seconds before lift off. NASA equipment detected an increased concentration of hydrogen before the three main engines ignited. While engineers at first said that the problem appeared to be a potentially dangerous build-up of hydrogen in the engine compartment, NASA officials later confirmed there was no indication of any fire in the shuttle, a faulted sensor reading caused the false alarm.

Just as numerous similar sensor health accidents, the above incident once again shows that the sensor data validation has become an essential part in the operation and control of modern industrial equipments, which rely completely on available sensor measurements.

On one hand, in order for a situation to be fully comprehended and controlled, reliable data acquisition and interpretation is of the utmost importance in modern control systems. Sensor data validation is therefore becoming an integral part of the modern decision-making processes.

On the other hand, an increasing functional control system makes it more complex and costly to accomplish the task of sensor validation and diagnosis. Major contributors to the increase are the sensor failure detection, identification and accommodation. While advances in integrated circuits continue to push down the cost of computing power, sensor technology has not kept pace so the relative cost of sensors has increased. Furthermore, in most cases, sensors must be built robust enough to withstand the harsh environment, which often account for the biggest single cost factor for a sensor.

In addition, control systems for critical plants, where operation must not be interrupted for safety reasons, are often configured with redundant sensors to provide fault tolerance and ensure the required degree of safety. Thus, the redundant sensors once again increase the whole system cost.

To achieve substantial savings of hardware redundancy, and, at the same time, to meet the requirements of reliable and accurate sensor measurements for the modern control systems, intelligent sensor data validation scheme was discussed in this thesis.

1.2 Concept of sensor validation

A sensor is an electrical or mechanical device that indicates characteristics of some form of energy impinging on them. [Manu95]

Sensor imprecision is an inherent property because sensors are subjected to noise. Justifications for the occurrence of noise are a poor understanding of the principles governing the sensor behavior, a lack of understanding of the environment, changing environmental conditions, tolerances within the sensor, and etc. Because of the sensor imprecision, automated sensor validation is needed for both real-time safety and post-test diagnosis. [NaSi98]

In the broadest sense, sensor validation is related to reliability, and is a determination that a sensor is or is not providing the correct signal. It also has other attributes such as diagnosis of the physical causes of the fault, a list of actions to take in priority in order to remedy the fault, and ways to ameliorate the fault by substitution, recalibration, or control action. In this thesis, most discussion will concentrate on sensor failure determination scheme.

1.3 Objectives of the thesis

The thrust of this thesis is on dealing with fuzzy-neural system based sensor validation scheme. To validate a signal, the sensor must be shown to consistently provide the true measurements, and the analysis of the validation hardware or software should trigger an alarm when the sensor signal significantly deviates from the correct value.

Neural network based models can be used to estimate critical sensor values when other neighboring sensors measurements are used as inputs. This framework is based upon the assumption that physically closed sensors are analytically related. [Brow92] The discrepancy between the measured and predicted sensor value may then be used as an indication of sensor health. The estimated sensor value may also be used as a synthetic data in the event of a sensor failure, often called soft sensor or inferential sensor.

Application of fuzzy set theory for the decision making process is advantageous, because the soft decision boundaries in fuzzy logic environments result in a flexible, more human-like decision. [Zade65] The information that the validation system deals with is often fuzzy rather than precise in nature, and fuzzy set theory allows us to model this imprecision appropriately and later permits us to reason in a linguistic language.

In this thesis, the sensor signal to be processed is a vibration signal. This kind of signal is one of the most common tools for detecting mechanical defects. The area of vibration signal validation has been considered difficult because vibration signals are influenced by factors such as noise, presence of multiple faults, severity of the fault, machine geometry, and speed variations. All these peculiarities make it difficult to vibration signal validation. [EsDi98]

To remedy this undesired situation, one can try to filter out the noise or use some kind of redundancy to back up the given sensor readings. Unfortunately, two (or more) sensor readings will never coincide. If they give readings in acceptable limits, it is not a big concern. However, when the readings are far apart, the signal from sensors need to be validated before they can be further used.

The standard approach to accomplish sensor validation is to use probabilistic means. In order to simplify the computation, probabilistic approaches commonly assume zero mean, Gaussian distribution of noise. This assumption is not always valid. Therefore, we propose to use fuzzy-neural system for sensor validation because no assumption on the noise characteristics needs to be made. Furthermore, unlike many probabilistic or knowledge based approaches in which the variance of the system perturbation must be known in advance, no such assumptions is made for this approach.

The fuzzy-neural system proposed in this thesis is called Winner Take All Expert (WTAE) network. As shown in Figure 1.1, the general idea of this algorithm is to divide a complicated problem into a series of sub-problems and assign a set of function approximators or 'experts' to each sub-problem. If a set of training cases may be naturally divided into subsets that correspond to distinct subtasks, interference can be reduced by using a system composed of several different 'expert' network with a Fuzzy Membership Clustering network that decides which of the experts should be used for each training case.

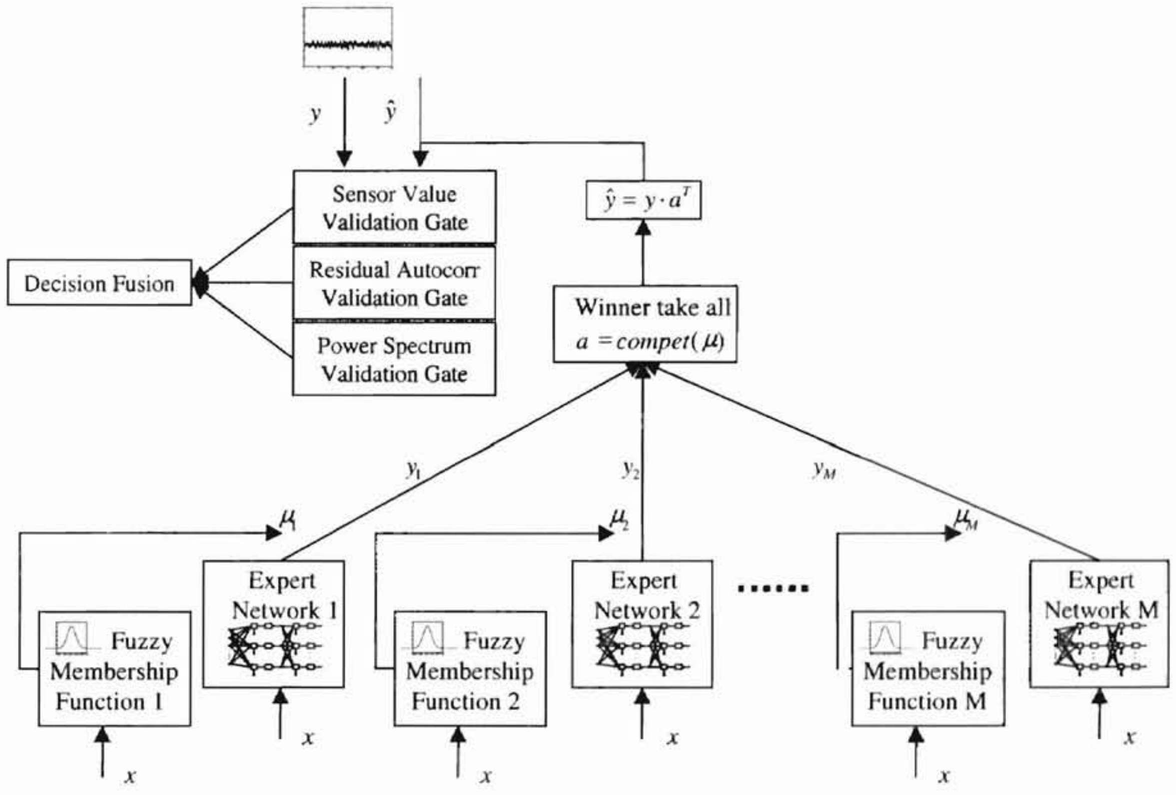


Figure 1.1: WTAE Network Architecture

In a similar spirit as the Hierarchical Mixtures of Experts networks [JoJa94], WTAE network works on the principle of 'divide and conquer'. The model employs growing fuzzy clustering method to divide the input space into overlapping regions on which 'experts' act. A simplification of the criterion leads to two joint criteria on the distance of the present pattern and the existing prototype centers in the input space and on the approximation error of the network for the given observation to be satisfied together. [KaNi93] A Fuzzy Membership Clustering network weights these experts' results to form a network output. The rational behind such a system is that the network allocates a new case to one expert, and, if the output is incorrect, the weight adaptations are localized to this single expert.

In learning stage, not only the relationship function of the sensors is built up, but also the characteristics of the critical sensor, both in time domain and frequency domain, are recorded for detection purpose. A time window is specified, and the variation of the sensor signal, residual autocorrelation, and power spectrum are all monitored and learned.

The signals from all the correlated sensors are fed to each expert of WTAE network simultaneously in the detecting stage. After the outputs of each expert are available, the Fuzzy Membership Clustering network will select the winner from them to form the tracking output.

Finally, the output from the estimator and the real sensor are compared both in time domain and frequency domain. Three fault indicators are used to identify the sensor failures. The first validation gate, sensor value validation gate, compares the tracking output directly with the real sensor data. In the second validation gate, the residual of the two time series is investigated by the autocorrelation coefficient. The power spectrum density of the two signals is compared in the last validation gate. A decision level fusion of the three validation gates is accomplished by the intersection of the three fuzzy sets. The output from the decision stage shows the sensor health confidence level of the critical sensor, which will provide as an indication for the human operator taking necessary actions in order to remedy or ameliorate the sensor faults.

1.4 Organization of the thesis

In Chapter 2 of this thesis a literature review of sensor validation algorithms is presented. Eight common sensor failure modes and the concept of sensor validation are described in the introduction part. Knowledge-based system, Kalman filter, neural network, and fuzzy logic based sensor validation schema are also discussed. Chapter 3 introduces the proposed WTAE network architecture and the validation algorithm. Chapter 4 shows the simulation results and comparisons to Multi-layer Perceptron (MLP) network estimator, Radial Basis Function (RBF) network estimator, and time domain fault indication scheme on benchmark problems. Finally, Chapter 5 provides the conclusion of the research and possible future work.

CHAPTER II

LITERATURE REVIEW

2.1 Sensors

A sensor is an electrical or mechanical device that indicates characteristics (presence, absence, intensity, or degree) of some form of energy impinging on it. Generally, it converts the energy (i.e. the physical stimulus) into electrical currents. We use these electrical signals for measurements or control purpose. [ErUp90]

There are all sorts of sensors used in different areas. [SmGa91] The accelerometers are used for measurement of vibration and acoustic emission. The sonar sensor generates a sound wave that spreads outward and is reflected back by a target, which can be analyzed for determination the range, bearing, and motion of the object. The pressure sensor uses the force applied by a diaphragm to a stack of quartz crystals to measure pressure. Radar is an electromagnetic sensor transmitting electromagnetic wave toward targets and observing the echoes returned. Many other sensors that derived from different technology, like optical sensor, mass flow, and thermocouple, measure all kinds of physical values. [Clar95]

2.2 Sensor failures

A sensor is declared faulty when it displays a non-permitted deviation from the characteristic properties of the objects it is monitoring. Sensor failure arises from every part of the sensor system, e.g. a blown-out amplifier in an accelerometer, or a loose connection in an optical sensor. Eight typical sensor failure modes [Yung92] can be identified: hard-over, bias, spike, stuck, erratic, cyclic, drift, and nonlinear. Figure 2.1 to 2.4 shows the characteristics of the eight sensor failure modes in time domain.

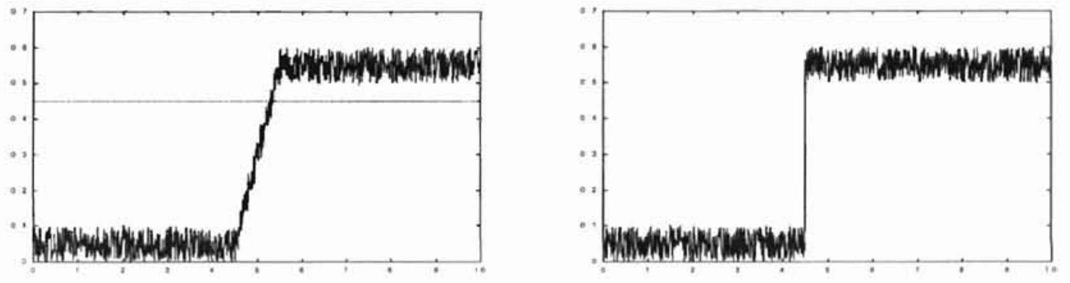


Figure 2.1: Hard-over and Bias Sensor Failure

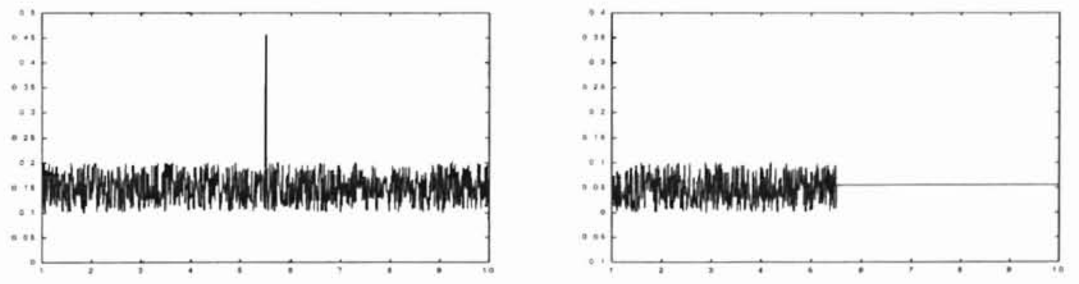


Figure 2.2: Spike and Stuck Sensor Failure

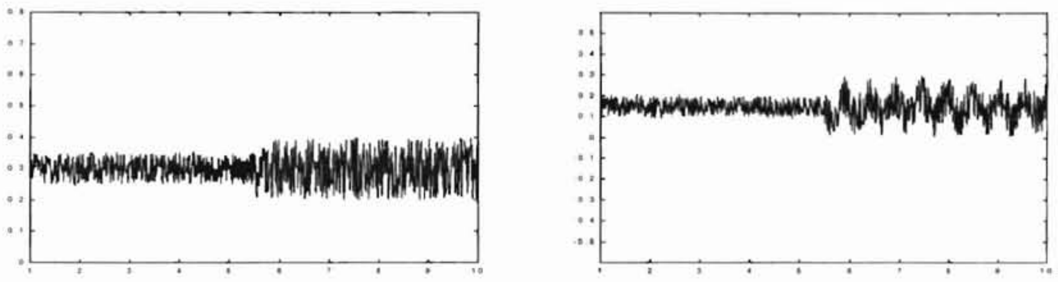


Figure 2.3: Erratic and Cyclic Sensor Failure

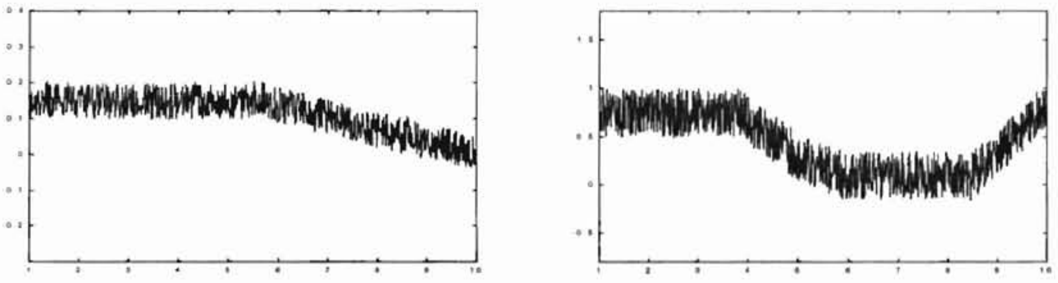


Figure 2.4: Drift and Nonlinear Sensor Failure

2.3 Sensor validation scheme

Sensor validation scheme is composed of three stages: commissioning stage, learning stage, and tracking stage. [YaCh97]

- In the commissioning stage, sensor specific data, like output limits, dynamic range and probable failure modes are provided by manufacturers and stored in an embedded memory. At the same time, validation information, including process time limit is decided during major process modification.

- Based on a period of failure-free observation, a time series model captures the sensor output characteristic in the learning stage. In most cases, this can be done by a stochastic Auto-Regressive Moving Average (ARMA) model. The unknown parameters in the model usually can be estimated by the Recursive Maximum Likelihood (RML) learning algorithm. In this stage, there is an assumption that the measurement noise is stationary and has a rational spectral density function (i.e. zero mean and uncorrelated). A modified ARMA model, Fuzzy Exponential Weighted Moving Average, was proposed in [Goeb96]. The parameter of the model was changed by fuzzy rules corresponding to system state, which improved the tracking performance.
- Sensor validation in the tracking stage will trigger the scheme into the alert phase when any detected abnormality occurs. [GoRi94] Usually, one or more fault indicators are used to indicate the level of difference between the tracking output and the real sensor signal.

2.4 Sensor validation based on hardware redundancy

Although tracking technology provides various methods for approximating sensor signal, hardware sensors are still the dominant references for sensor validation in modern control system.

In most cases, validation scheme uses a number of hardware sensors provided a redundancy of information monitoring each important system datum, from which a more reliable value was extracted. There are three different techniques used in this area. [SmGa91]

- More than one sensors of the same kind monitor the same scenario. In this case, any disagreements among sensors can be solved statistically for the overall data values.
- Providing different kind of sensors in proximity to monitor the same datum. The advantage of this situation from the above is minimizing the possibility of common fault problems of same kind sensor. For example, to measure the temperature, half sensors might be thermocouples while the other half might be mechanical temperature-sensors.
- Using different types sensors in different locations to infer the same target. In this situation, the values can be obtained not only directly but also from other system data based on well-established analytical relationship. In particular, the redundancy obtained through a functional relationship is often referred to as an analytical redundancy [Lee94].

While hardware redundancy is popular and solves many sensor validation problems, it has some disadvantages:

- The expense of the redundant sensors, and the installing, maintaining of them for each important observation greatly increase the cost of the system.
- Common fault failures still happen to all sensors.

2.5 Knowledge based system

The technology from Artificial Intelligence (AI) helped to establish Knowledge Based System (KBS) that diagnose sensors utilizing system operational knowledge in validating sensor values. [Lee94]

To find out a potential fault from the observed sensor, it is necessary to derive the normal expected values of the sensor under current operating environment. The relationships existing among the sensors are defined as the Causal Relations (CR). Consequently, a chain of several CRs can be realized by a data structure called Sensor Redundancy Graph (SRG). CR represents the vertex in an SRG, and two CRs having any common sensor are directly connected through an edge. Any sensor belonging to a SRG can be used to validate others in the same SRG and vice versa.

An example of SRG is shown in Figure 2.5. Each R_n represents a CR. Two CRs having any common sensors are connected through an edge F_m.

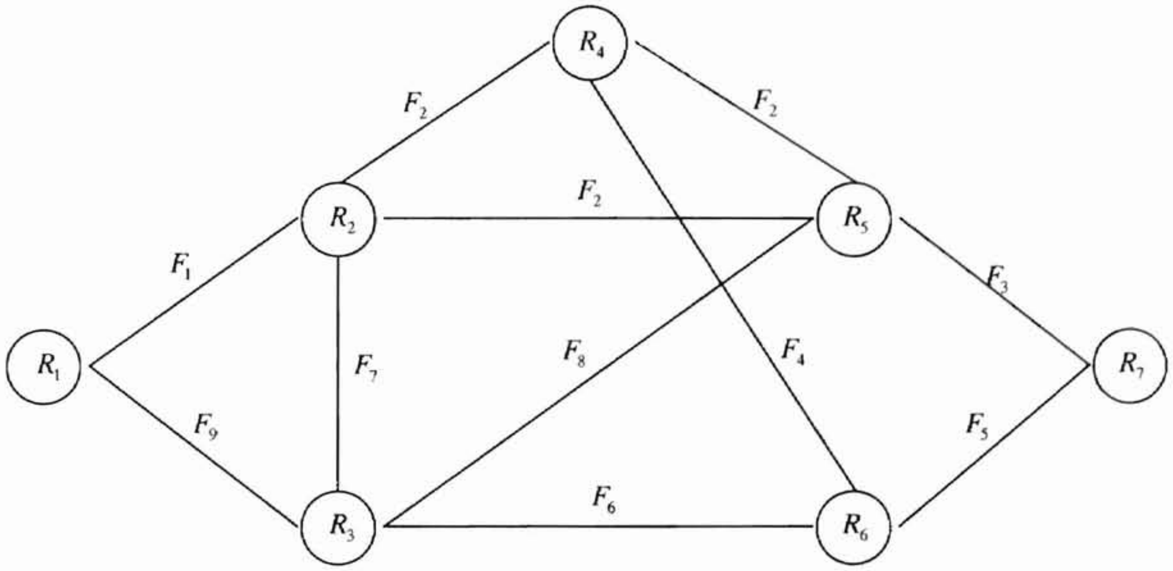


Figure 2.5: An Example of SRG

A CR can be expressed as an equation or a table, and can be obtained by physical laws, system design, and operational data. Because the fault source more likely affects the sensors nearby, a CR should be formed with sensors that are close together along system configuration.

The consistency index of a relationship, R_i, is denoted as $[R_i]$

$$[R_i] = \begin{cases} 1, \text{ if } R_i \text{ consistent.} \\ 0, \text{ if } R_i \text{ not tested.} \\ -1, \text{ if } R_i \text{ not consistent.} \end{cases} \tag{2.1}$$

An inconsistent CR can be made consistent if it becomes consistent with any sensor substituted from adjacent CRs.

The Validity Level (VL) of a sensor is denoted as $[S_k]$. It is i level if the summation of the consistent index of all CR supporting this sensor is number i, or it is supported by a set of sensors whose minimum validity is i.

The knowledge-based system is efficient and helpful to yield valuable clues to the fault diagnosis and their locations. But it requires accurate models with a comprehensive understanding of the basic physics and the knowledge of all the potential variables, which are often unavailable in most cases.

2.6 Soft sensor used in tracking stage

A soft sensor [Gonz94], or virtual sensor, is a system designed to substitute the momentary or permanent unavailability of a sensor in a plant. For sensor validation purpose, it is compared with the real output of the sensor to detect the sensor failure modes. Several aspects have to be considered in establishment of soft sensor model:

- A soft sensor model is different from the control model. The control model is required to predict a signal within a relatively short period, while in the case of soft sensor model a larger time prediction span is usually necessary.
- The larger the part of plant where soft sensor derived from, the more complex the model will be, and the time required to train the parameter may be much longer. So a model using only a small part of the plant would be desirable.

The main tracking algorithms used in this area are Kalman Filter and Extended Kalman Filter. The fundamental assumptions of the Kalman Filter are that the linear plant equations are known and has zero mean white noise with known covariance. These assumptions make this approach difficult to model nonlinear system and vulnerable to correlated noise. The Extended Kalman Filter tries to linearize the nonlinear functions, but it is sensitive to the accuracy of the initial conditions. [Goeb96]

2.7 Neural network for sensor validation

Neural network can be proposed as a tracking algorithm for sensor validation. It has the following properties [NaWi98]:

- Applicability to Nonlinear System: It was shown that neural network with one hidden layer could uniformly approximate any continuous function. [Horn89]
- Parallel Distributed Processing: These architectures have high degree of fault tolerance and high processing speed, due to the simplicity of the computations.
- On-line learning and Adaptation: Network can be trained by both prerecorded data and current data.

- Applicability to multivariable systems: Neural network can be Multi-input Multi-output (MIMO), and this naturally leads to their applications to multivariable systems. [GuNu91]

Neural network based models can be used to estimate critical sensor values when other sensor measurements are used as inputs. The basic premise behind this framework is that the sensed plant variables are not independent of each other. [Brow92]

The most popular neural network in use today is the Multi-layer Perceptron (MLP) network. The feasibility of using MLP network in sensor validation [ErUp90] [GuNu91] has been studied since 1990. And a modified MLP network [NaSi98] was implemented for hardware based on-line learning soft sensor in 1998.

A Kalman Filter based sensor validation scheme and an on-line learning MLP network were compared in [Marc98]. On one hand, the Kalman Filter based scheme is good at the robustness capabilities. On the other hand, the neural network scheme has potential for on-line learning, particularly in the case of poorly modeled dynamics.

However, MLP network is easily getting stuck in local minima on the error surface in training stage. Yet, there is no efficient method available for determining a minimal and adequate architecture for a given problem.

A Radial Basis Function (RBF) neural network was investigated in [WhDh94] for sensor signal tracking. The network used the K-Means clustering algorithm for placement of the basis function centers. The result showed that RBF network is good at tracking the sensor signal when it changes slowly. But a general RBF network could not accurately learn the data and became unstable when the problem becomes more complex.

In Summary, neural network has been studied for tracking sensor signal in sensor validation area for about ten years. Traditional neural network such as Multi-layer Perceptron (MLP) network and Radial Basis Functions (RBF) network have been proven successful as universal function approximators. But the training algorithm is typically too slow for solving real world problems. In addition, when the problem becomes complicated, most of the tracking network even could not even converge to a local minimum in a reasonable time due to the limitation of the hardware and inefficiency of the learning rule.

It was concluded in [HaLi98] that if a neural network could not be established to learn the relationship from the available data, the reason will be one of the following:

- Insufficient training data,
- No relation among the variables,
- Inappropriate architecture of the neural network, or
- Inappropriate training algorithm.

If the problem is caused by the first two reasons, there's nothing we can do by using neural network. Therefore, most research efforts in this area were made on the searching for appropriate structure and modified learning algorithm.

2.8 Time domain indicators used in detecting stage

Five failure alarm indicators are used in literature to classify the aberrations observed: limit indicator, jump indicator, mean indicator, noise indicator, and drift indicator. [Yung92]

- The limit indicator simply tracks the raw sensor data and its derivative exceeding the physical limit of sensors and the operating environment. It primarily targets for the hard-over failure mode.
- The spike and bias failure modes can be detected by the jump indicator and the mean indicator, which monitor the excessive deviation from the data's mean level. When the mean changed significantly, and jump happens, the failure mode is in a bias mode. While the time series jump without significant mean change, it is spike.
- The noise indicator follows the variance of the time series. When the noise is low, a high variance indicates erratic fail, and a low variance below the minimum variance is a symbol of stuck. Under high noise environment, the influence is analyzed to check the cyclic failure mode.
- The last indicator, drift indicator, registers the significance and the persistence of drifting actions.

A series of simulation results showing eight failure modes detected by time domain indicators are shown in Figure 2.6.

The Combination of all alarm indicators in time domain provides an efficient and cost effective method to analyze the sensor failure modes. But the detecting system needs sufficient information of both sensor characteristics and environments feature, which in most cases are unavailable or insufficient. In addition, when the amplitude of the fault signal is not big enough to trigger the 'Noise indicator', it is impossible for time domain indicators to detect and identify some of the failure modes.

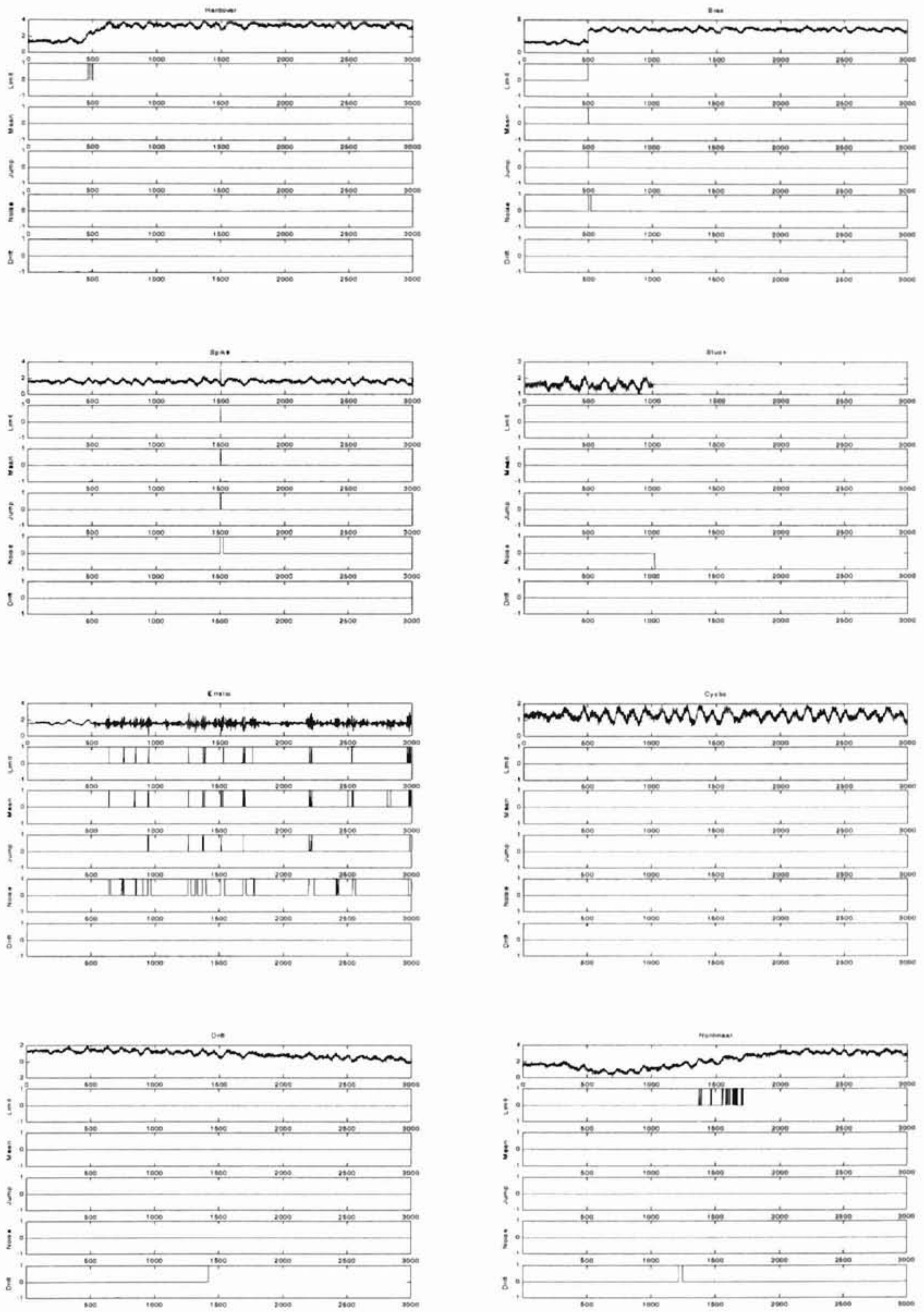


Figure 2.6: Simulation Results of Time Domain Indicators

2.9 Fuzzy sensor validation gate

A new algorithm for fuzzy sensor validation was described in [Goeb96]. The redundancy situation is more than one sensor of the same kind monitoring the same operating environment.

In the detecting stage, a dynamic validation curve based on fuzzy membership technology was introduced for each involved sensor. After the validation, the validity levels, or confidence levels from the results of each validation gate were combined for data level sensor fusion. This fused value is then used for prediction of the next possible sensor output, which in turn is necessary to perform the validation of the next step. The confidence level of the critical sensor depends on the specific sensor characteristics, the predicted value, and the physical limitations of the sensor signal variation.

A choice for validation curves $v(z)$ for a particular situation could be a piecewise bell curve of the form $v(z) = e^{-\left(\frac{\hat{x}-z}{a_w}\right)^2}$, where a_w has to be chosen separately for the left curve and the right curve.

An example formula of validation gate could be:

$$\sigma = \begin{cases} 0 & z < v_{left} \\ \dfrac{e^{-\left(\frac{\hat{x}-z}{a_{left}}\right)^2} - e^{-\left(\frac{\hat{x}-v_{left}}{a_{left}}\right)^2}}{1-e^{-\left(\frac{\hat{x}-v_{left}}{a_{left}}\right)^2}} & v_{left} < z \le \hat{x} \\ \dfrac{e^{-\left(\frac{\hat{x}-z}{a_{right}}\right)^2} - e^{-\left(\frac{\hat{x}-v_{right}}{a_{right}}\right)^2}}{1-e^{-\left(\frac{\hat{x}-v_{right}}{a_{right}}\right)^2}} & \hat{x} < z \le v_{right} \\ 0 & z > v_{right} \end{cases} \tag{2.2}$$

where

σ is the confidence value for a particular sensor,

v_{right} and v_{left} are the right and left validation gate borders, respectively,

a_{right} and a_{left} are the parameters for the left and right validation curve, respectively,

z is the sensor reading, and

$\hat{x}$ is the predicted value.

Fuzzy rules were used to determine the shape of the validation curve. The curve changes dynamically with the operating conditions, which allow capturing the change in sensor behavior. Other effects, like environmental conditions, can be integrated into the curve as well.

The fuzzy validation scheme was shown working very well in the unknown noise type environment and non-symmetric noise distributions. But it performed slightly worse compared to the Kalman filter in the presence of Gaussian noise.

CHAPTER III

NETWORK ARCHITECTURE AND DETECTING ALGORITHM

3.1 Multi-layer Perceptron (MLP) neural network

Artificial neural network has obtained attention recently because they seem to resemble functions which are similarly performed by biological neurons. A network of artificial neurons is a data processing system consisting of a large number of simple, highly interconnected processing elements in architecture inspired by the structure of the cerebral cortex portion of the brain. [Ande72]

The first description of Multilayer Perceptron (MLP) neural network was introduced in the thesis of Paul Werbos in 1974 [Werb74]. It was not until the mid 1980s that the back-propagation training algorithm was rediscovered and widely publicized. It was independently developed by David Rumelhart, Geoffrey Hinton and Ronald Willeams [RuHi86], David Parker [Park85]. The multi-layer perceptron neural network, trained by the back-propagation algorithm, is currently the most widely used neural network.

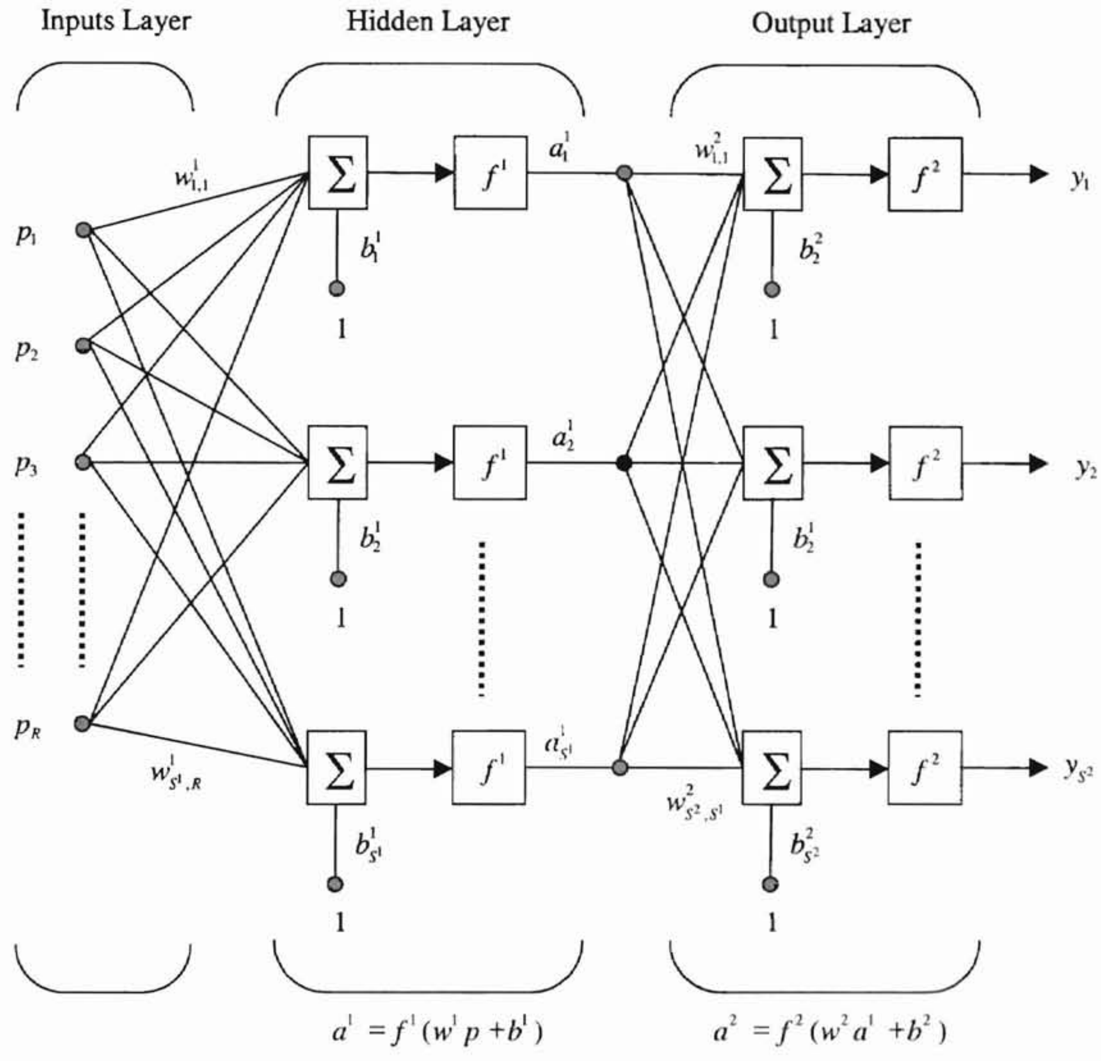


Figure 3.1: Topology of a Three-layer MLP Neural Network

During training of the MLP network, the information flows forward from the input layer to the output layer, and the error, which is the difference between the target and the output of the network, is propagated backwards through the weights to calculate the weight adjustments. Training involves the modification of the weights until the error is reduces to an acceptable limit. The first layer receives the input, modifies it using the set of weights, and passes it to the hidden layer; each hidden layer in turn propagates the modified inputs to the next layer and eventually to the output layer where the overall error is calculated. The hidden layers are used to represent nonlinear characteristics of the data set. The architecture of a typical three-layer MLP network is shown in Figure 3.1.

The gradient descent with momentum training algorithm is often too slow for practical problems. In this thesis we employ a heuristic techniques, Resilient Back-propagation [RiBr93], which can converge from ten to one hundred times faster than the standard steepest descent algorithm.

The majority of adaptive algorithm performs a modification of learning rate according to the observed behavior of the performance function. What is often disregarded is that the size of the actually weight update step Δw_{ji}^{m} is not only depend on the learning rate, but also on the partial derivative $\frac{\partial E}{\partial w_{ji}^{m}}$. MLP network typically use sigmoid transfer function in the hidden layers. Sigmoid functions are characterized by the fact that their slope will approach zero, as the input gets large. This causes small changes in the weights and biases, even though the weights and biases are far from their optimal values.

In order to determine the size of the weight-update, we define individual update-value Δ_{ji} for each weight. The learning rules are following:

$$\Delta_{ji}^{m}(k+1) = \begin{cases} \eta^{+}\Delta_{ji}^{m}(k) & , \quad if\ \frac{\partial E(k)}{\partial w_{ji}^{m}}\frac{\partial E(k-1)}{\partial w_{ji}^{m}} > 0 \\ \eta^{-}\Delta_{ji}^{m}(k) & , \quad if\ \frac{\partial E(k)}{\partial w_{ji}^{m}}\frac{\partial E(k-1)}{\partial w_{ji}^{m}} < 0 \\ \Delta_{ji}^{m}(k) & , \quad else \end{cases} \tag{3.1}$$

where $0 < \eta^{-} < 1 < \eta^{+}$.

Every time the partial derivative of the corresponding weight w_{ji}^{m} changes its sign, which indicates that the last update was too big and the algorithm has jumped over a local minimum, the update-value Δ_{ji} is decreased by the factor η^{-}. If the derivative retains its sign, the update-value is slightly increased in order to accelerate convergence in shallow regions.

Besides the convergence speed, there are a number of practical concerns when using MLP network. The first is what size network best for a given problem. With little or no prior knowledge of the problem, one must determine the network size by trial and error. A methodical procedure is recommended. One approach is to start with the smallest possible network and gradually increase the size until the performance begins to level off.

The second is the time complexity of learning. That is, we may ask if it is possible to learn the desired mapping in a reasonable amount of time. If we have a very large problem, e.g. if the dimension of the input space is very large, or the relationship between input and output is too complex, then it is unlikely that we will be able to determine if a weight solution exists in a reasonable amount of time. [BlRi88] On the other hand, learning algorithm like Back-propagation are based on a gradient search, which is a greedy algorithm that seeks out a local minimum and thus may not yield the exact mapping. Usually it is dangerous to increase the learning rate, because the algorithm may be unstable when it reaches the steep parts of the surface. But there are still several somewhat successful simple gradient search algorithms. Most of them introduce additional parameters which are difficult to determine, and must be varied depend on the characteristics of the problems.

3.2 Radial Basis Function neural network

The Radial Basis Function Neural Network (RBF) learns the mapping from a set of data in the form of input-output observation pairs (x_n, y_n). In sensor validation problem, x_n is a M-dimensional input vector and y_n is an output scalar. The n^{th} observation can be described as:

$$I^{(n)} = \{(x_n, y_n) : x_n \in D \subseteq R^M ; y_n \in R\} \tag{3.2}$$

where D is a subset of the space of all real valued M-dimensional vectors R^M. The observation $I^{(n)}$, $n = 1,\ldots,N$ are assumed to be consistent with an underlying function:

$$f^*(x_n) = y_n . \tag{3.3}$$

The mapping described by RBF is denoted by:

$$f : x \to y(D \to R) . \tag{3.4}$$

The single hidden layer RBF linearly combines the output of the hidden units. Each of the hidden units construct a mapping and hence these mapping can be viewed as the basis functions Φ_k, $k = 1,\ldots,K$, the total number of hidden units. The output is thus represented as:

$$f(x) = \sum_{k=1}^{K} \alpha_k \Phi_k(x) , \tag{3.5}$$

where α_k is the weighs in the output layer.

The most common Radial Basis Functions are:

$$x \to y_1 = f_1(x, w^1) = \Phi_1(x),\ldots, x \to y_k = f_k(x, w^K) = \Phi_k(x) , \tag{3.6}$$

where

$$y_k = f_k(x, w^k) = \exp[-\|x - w^k\|^2 /(2\sigma_k^2)] = \Phi_k(x), \; k = 1,...K \qquad (3.7)$$

with center at w^k, and σ_k representing the required smoothness. The parameter σ_k is the spread of the RBF representing its span around w^k in the input space.

A typical Radial Basis Function with $w^k = [12,12]^T$ and $\sigma_k = 3$ in three dimensions is shown in Figure 3.2:

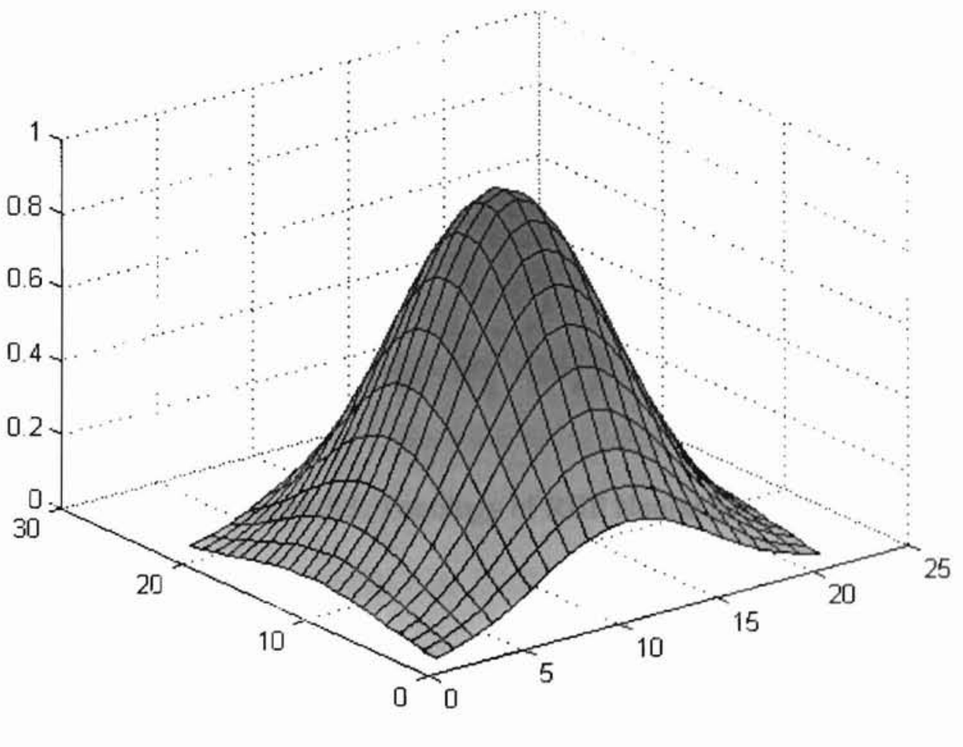


Figure 3.2: Typical Radial Basis Function

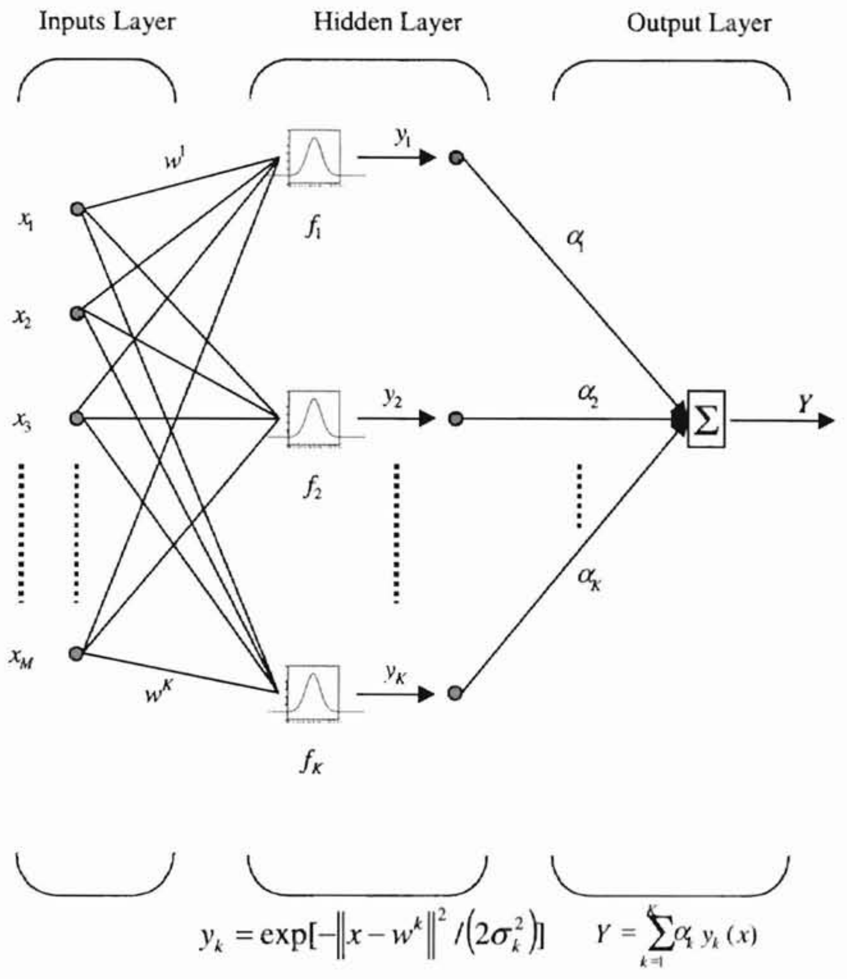


Figure 3.3: Topology of RBF Neural Network

Each RBF of the neural network shown in Figure 3.3 is influential only on its receptive field, which is a small region of the input space R^M. The important regions of the input space, where training samples are covered jointly by the K RBFs, are centered on the clusters that represent subclasses.

The training is broken down into two stages: [PaSa91] learning in the hidden layer, followed by learning in the output layer. Learning in the hidden layer is typically performed using a clustering algorithm. A popular choice is the K-means algorithm. It requires the number K of clusters to be given, $K < Q$, where Q is the total number of the training samples. The process begins by assigning the first K sample input vector $x_1,...x_K$ to be the center $w^1,...w^K$, respectively. Then, it assigns each of the left $Q-K$ input vectors $x_{K+1},...x_Q$ to the cluster it is closest in Euclidean distance. The input vectors for each k^{th} cluster are averaged to determine a new cluster center w^k. Next, each of the Q input vectors is again assigned to the class to whose new representative center it is closest. This loop is repeated until no clusters change any more.

The variance σ_k^2 for each cluster k is determined by:

$$\sigma_k^2 = \frac{1}{n(k)} \sum_{\{q:class(q)=k\}} \left\| x_q - w_k \right\|^2, \tag{3.8}$$

where $n(k)$ is the number of vectors in each cluster k.

Learning in the output layer is performed after the parameters of the basis functions have been determined. That is after learning in the hidden layer is completed. The output layer is typically trained using the Least Mean Squares algorithm. Here, we will use the Moore-Penrose algorithm for numerical convenience.

The RBF can be represented in state space form:

$$Y = \Phi \cdot \alpha, \tag{3.9}$$

where,

$$\alpha = [\alpha_1,...\alpha_K]^T,$$

and

$$\Phi = \begin{bmatrix} \exp[-\left\| x_1 - w^1 \right\|^2 /(2\sigma_1^2) & \cdots & \exp[-\left\| x_1 - w^K \right\|^2 /(2\sigma_K^2) \\ \vdots & \ddots & \vdots \\ \exp[-\left\| x_Q - w^1 \right\|^2 /(2\sigma_1^2) & \cdots & \exp[-\left\| x_Q - w^K \right\|^2 /(2\sigma_K^2) \end{bmatrix}. \tag{3.10}$$

The weights in the output layer then can be calculated with the Moore-Penrose inverse:

$$\alpha = \left(\Phi^T \cdot \Phi + \mu I\right)^{-1} \cdot \Phi^T \cdot Y\,, \tag{3.11}$$

where μ is a constant less than 1, and I is the identity matrix.

One of the major advantages of the RBF network is that learning tends to be faster than in MLP. [WhDh94] Once the hidden layer parameters are fixed, learning in the output layer parameters is easier because the network output is linear in weights. The most difficult part is the training in the hidden layer. When a cluster had a population, which was small in comparison to the neighborhood size, the variance of that cluster could be made so large that significant error occurred. On the other hand, when a cluster had a population that was great in comparison to the neighborhood size, the variance could be made so small for that cluster that inadequate generalization occurred. [XuOj93]

3.3 Fuzzy logic basis

In 1965, Zadeh introduced tools for working with fuzzy natural language terms, which he called "fuzzy logic". [Zade65] Terms used in natural language to describe some concepts with vagueness are called linguistic variable such as "small", "medium", and "large". To further refine these variables, adverbs such as "very" or "somewhat" can be added at will.

The ability to reason in imprecise terms is a great advantage especially when solving complex tasks. In Boolean or classical logic, each statement has a value "true" or "false". In fuzzy logic, as in some types of multi-valued logic, the transition from membership to non-membership of elements in the set is gradual rather than abrupt, and the membership can take any value in the closed interval [0,1]. Using fuzzy set theory as the basis of inference establishes an estimated foundation for obtaining an accurate form that carries out the condition of inexact rather than exact rationale. The main advantage of fuzzy logic lies in the face that they provide a soft decision, a value that describe the degree to which a pattern fits within a class.

One should differentiate between randomness and fuzziness. Randomness has to do with uncertainty of an object in a crisp subject. Fuzziness is associated with the fact that classes do not have sharp boundaries, and that there is no sharp transition between membership and non-membership. The theoretical foundation of the fuzzy logic has a well established mathematical framework. Thus, the source of imprecision is not the theory itself, but rather the manner in which linguistic variables are applied to the formulation of a real world problem. [ZhKa94]

We use fuzzy set membership functions to assign fuzzy truths to proposition involving linguistic variables. A membership function is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The bell-shaped (Gaussian) fuzzy set membership function shown in Figure 3.2 is a convenient form defined by:

$$f(x) = \exp[-(x-c)^2 / 2\sigma^2] \tag{3.12}$$

which depends upon the two parameters c and σ^2.

Fuzzy logic operators, used to combine two or more fuzzy sets to produce a single set, are defined by a function:

$$f : [0,1]^n \rightarrow [0,1],\ n \geq 2, \tag{3.13}$$

Where n is the number of aggregation sources. The basic fuzzy logic operators are intersection and union.

The union of two fuzzy sets is defined as

$$\mu_{A\cup B}(x) = \max[\mu_A(x), \mu_B(x)], \tag{3.14}$$

while the intersection is defined as

$$\mu_{A\cap B}(x) = \min[\mu_A(x), \mu_B(x)]. \tag{3.15}$$

Often a crisp value is necessary to interact with a real environment which is accomplished in the defuzzification step. One popular method is the centroid method:

$$y_j = \frac{\sum_{i=1}^{P} y_i \mu(y_i)}{\sum_{i=1}^{P} \mu(y_i)}, \tag{3.16}$$

where, y_j is the overall value, and y_i is the value according to rule i with membership value, $\mu(y_i)$.

It is proven in [WaMe92] that fuzzy system with Gaussian fuzzy set membership function is a universal appximator on compact sets. It implies that for normalized features, the feature vectors in the hypercube $[0,1]^N$ can be mapped to any output identifier vectors as closely as desired, provided that there are sufficient number of fuzzy sets and fuzzy rules.

3.4 Introduction to WTAE network

Traditional neural network such as Multi-layer Perceptron Network and Radial Basis Functions Network have proved successful as universal function approximators and have been used in various problems. But the training algorithm is typically too slow for solving real-world problems in real time. In addition, when the problem becomes complicated, most of the systems could not even converge to a local minimum in a reasonable time due to the limitation of hardware and the inefficiency of the learning rule. Motivated by such concerns, a number of researchers have investigated methods of function approximation incorporating ideas from the communities of both statistics and neural network. [ChPu88] The general approach is to divide a complicated problem into several simpler sub-problems and assign a set of function approximators or 'experts' to each sub-problem. [JoJa94]

If a set of training cases may be naturally divided into subsets that correspond to distinct subtasks, interference can be reduced by using a system composed of several different 'expert' network plus a fuzzy membership clustering network that decides which of the experts should be used for each training case. [FePu98] A system of this kind can be used only when the division into subtasks is known prior to training.

The proposed Winner Take All Experts network architecture is a modular architecture that works on the principle of 'divide and conquer'. The model employs growing fuzzy clustering method to divide the input space into overlapping regions on which 'experts' act, and a fuzzy membership clustering network to weight these experts to form a overall network output. The idea behind such a system is that the growing fuzzy clustering algorithm allocates a new case to one of the experts, and, if the output is incorrect, the weight adaptations are localized to this expert.

3.5 The Winner Take All Experts (WTAE) network architecture

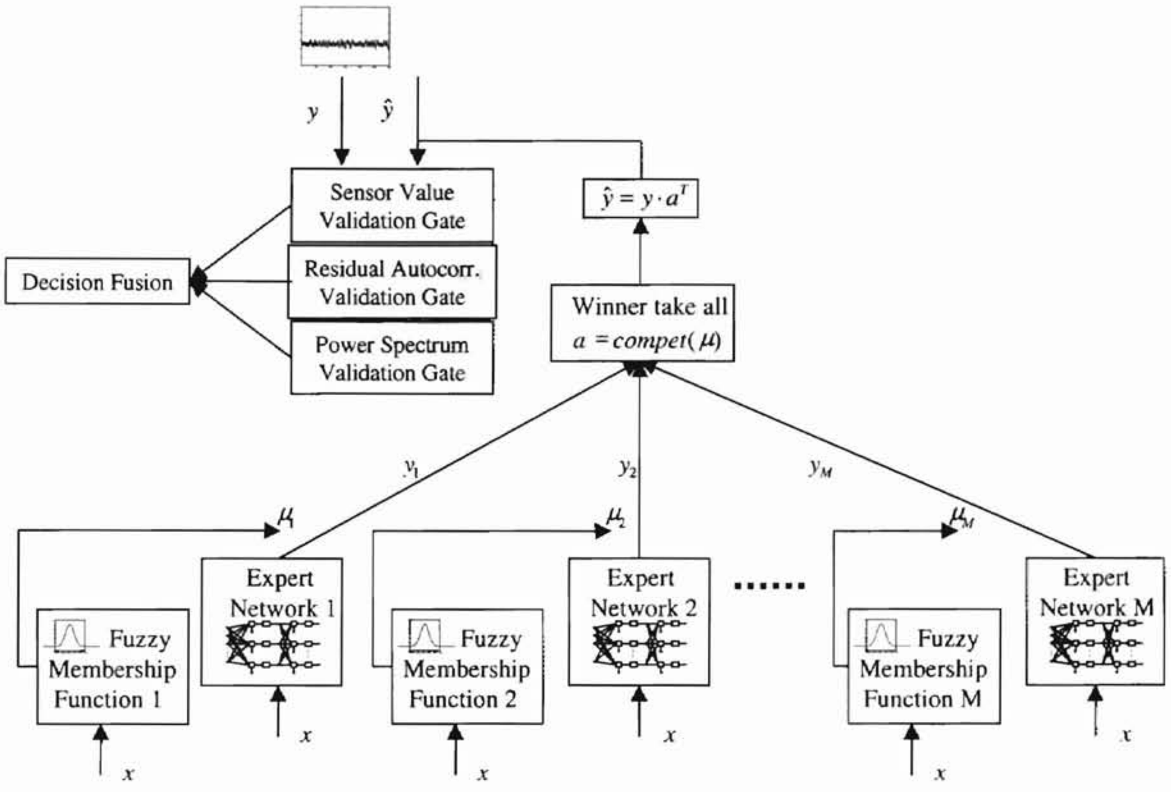

y
$\hat{y}$
Sensor Value Validation Gate
Residual Autocorr. Validation Gate
Power Spectrum Validation Gate
Decision Fusion
$\hat{y} = y \cdot a^T$
Winner take all $a = compet(\mu)$
y_1
y_2
y_M
μ_1
μ_2
μ_M
Expert Network 1
Expert Network 2
Expert Network M
Fuzzy Membership Function 1
Fuzzy Membership Function 2
Fuzzy Membership Function M
......
x

Figure 3.4 WTAE Network Architecture

The proposed Winner Take All Experts Network model is shown in Figure 3.4.

In the given architecture, each expert network is a typical single hidden layer MLP network. The network input vector x consists of the signals from each related sensors that located nearby the sensor of interest. The outputs from every sub-network form an output vector $y = [y_1, y_2, ..., y_M]$.

The vector x is simultaneously fed into the Fuzzy Membership Clustering network, which produces membership level, $\mu_i(x) \in [0,1]$, where $i = 1,...,M$. Each membership level corresponding to each sub-network altogether forms a membership vector $\mu = [\mu_1, \mu_2, \ldots, \mu_M]$.

The fuzzy membership clustering network is a single layer network with a Gaussian output non-linearity. Let the i^{th} center and the i^{th} variance of the fuzzy membership level network be $C_i = [C_{i1}, C_{i2}, \ldots, C_{iN}]^T$ and $\sigma_i = [\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{iN}]^T$ respectively, where N is the dimension of input space and 'T' denotes the transpose operation. The corresponding output $\mu_i(x)$ after the Gaussian non-linearity is given by:

$$\mu_i(x) = \prod_{j=1}^{N} e^{-\frac{(x_j - C_{ij})^2}{2\sigma_{ij}^2}} . \tag{3.17}$$

The Gaussian function is to ensure that

$$\mu_i \in [0,1],$$

and

$$\|x - C_i\| < \|x - C_j\| \Leftrightarrow \mu_i > \mu_j, \tag{3.18}$$

where $j = 1,2,\ldots,i-1,i+1,\ldots,M$, and M is the number of total experts.

Thus, the i^{th} expert's influence is localized to a region around C_i.

A Winner Take All function is denoted by as $a = compet(\mu)$, which takes one input argument μ, and returns output vector with 1 at the ith element having maximum membership value ($i = \arg\max_j (\mu_j(x))$), and 0 elsewhere. The Winner Take All function selects the winner expert network outputs to form the overall outputs, $\hat{y}(x) = y \cdot a^T$.

Finally, the output from the estimator and the real sensor are compared both in time domain and frequency domain. Three fault indicators are used to identify the sensor failure. The first validation gate, sensor value validation gate, compares the tracking output directly with the real sensor data. In the second validation gate, the residual of the two time series is investigated by the autocorrelation coefficient. The power spectrum density of the two signals is compared in the last validation gate. A decision level fusion of the three validation gates is accomplished by the intersection of the three fuzzy sets. The output from the decision stage shows the sensor health confidence level of the critical sensor, which will provide as an indicator for the human operator taking necessary actions in order to remedy or ameliorate the sensor fault, if any.

3.6 Training algorithm

The Winner Take All Experts network is trained using Growing Fuzzy Clustering algorithm, where the data are trained sequentially. The network will continuously add experts that contribute to the final estimate in off-line training step. The growth criterion is so important that if the new expert contributes little to the output, then, not only does the complexity of the network increase unnecessarily, it adds to the computational burden. This leads to the question of how the network growth must be limited.

Figure 3.5 shows us the training steps used in the WTAE network, where the training data are received sequentially. The network begin with no hidden unit. The first observation (x^1, t^1), where x^1 is the input and t^1 is the corresponding target output, is used in initializing the first fuzzy membership center $C_1 = x^1$. And the initial variance for the first cluster is set as:

$$\sigma_{11} = \sigma_{12} =, \dots = \sigma_{1N} = 1, \tag{3.19}$$

where N is the number of input dimension. Now, we get our first fuzzy membership center as:

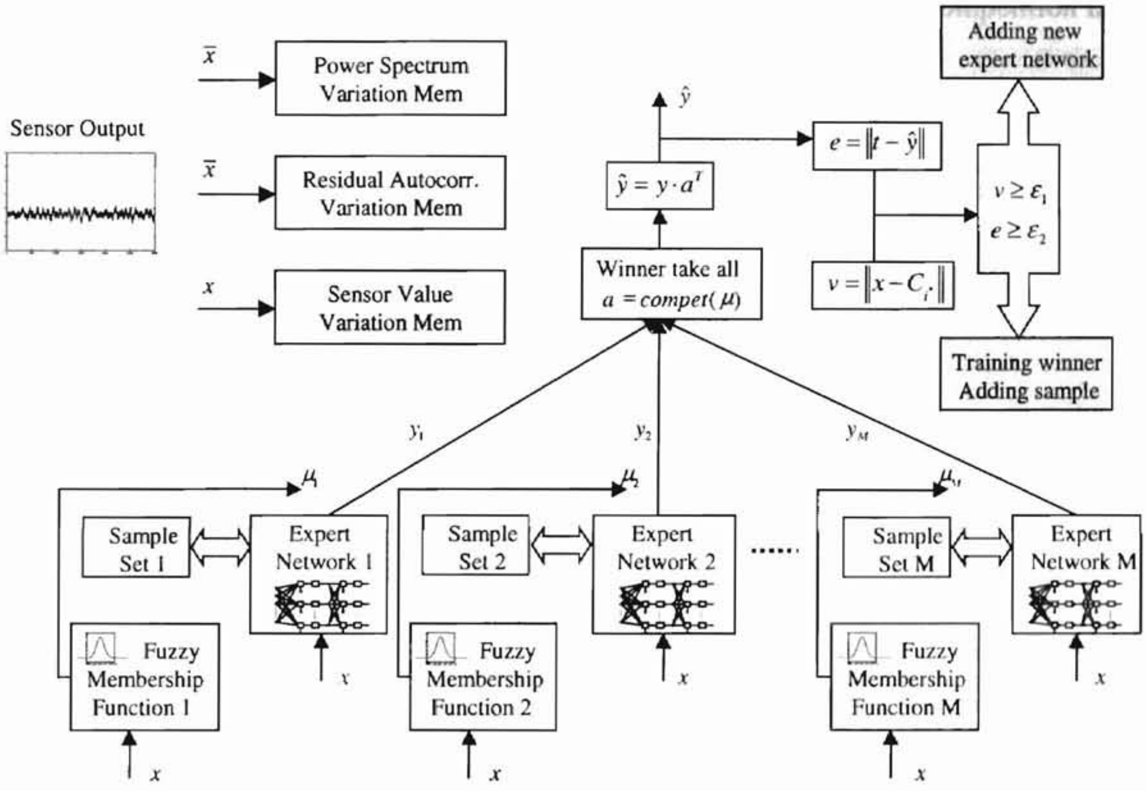


Figure 3.5: WTAE Network Training Algorithm

$$\mu_1(x^1) = \prod_{j=1}^{N} e^{-\frac{(x_j^1 - C_{1j})^2}{2\sigma_{1j}^2}} . \tag{3.20}$$

At the same time the first sample is stored locally in the first training set corresponding to the first fuzzy center defined above.

Next, a simple architecture MLP expert is built up and trained by the first training set. The structure of the MLP is determined by the complexity of the problem. The training goal, which decides the stopping criteria for the MLP training, is determined by the maximum difference between estimator and the real sensor output (target) we can tolerate.

The overall output is calculated based on winner take all rule. First, the whole output μ from fuzzy clusters goes through the competition layer where each neuron excites itself and inhibits all the other neurons. The transfer function of competition layer is defined as following:

$$a = compet(\mu), \tag{3.21}$$

where $\mu = [\mu_1, \mu_2, ..., \mu_M]$. It works by finding the index i^* of the neuron with the largest net input, and setting its output to 1 (with ties going to the neuron with the lowest index). All other outputs are set to 0.

$$a_i = \begin{cases} 1, & i = i^* \\ 0, & i \neq i^* \end{cases}, \text{ where } \mu_{i^*} \geq \mu_i, \ \forall i \neq i^*. \tag{3.22}$$

Next, the overall output $\hat{y}$ is calculated as following:

$$\hat{y} = y \cdot a^T, \tag{3.23}$$

where $y = [y_1, y_2, ..., y_M]$, the outputs from every expert. Actually, it is unnecessary to go through every expert. After the winner is selected by fuzzy membership function, the only MLP has to be calculated is the winner expert.

As observations are received, the network grows by adding new experts. The decision to add a new expert for an observation (x^k, t^k) depends on its novelty, for which the following two conditions must be met:

$$\left\| x^k - C_{i^*} \right\| \geq \varepsilon_1 \tag{3.24a}$$

$$e^k = \left\| t^k - \hat{y}^k \right\| \geq \varepsilon_2 \tag{3.24b}$$

where C_{i^*} is the nearest fuzzy center to x^k in the input space, $\hat{y}^k$ is the resulted output from WTAE network, and $\varepsilon_1, \varepsilon_2$ are thresholds. The first criterion decides that the input must be far away from the existing fuzzy centers, and the second criterion says that the error between the network output and the target value must be significant. The criterion ε_1 represents the scale of resolution in the input space. On the other hand, the value ε_2 is chosen to represent the desired accuracy of the network output. [WaLo96]

When a new expert is added to the network, the parameters associated with this expert are assigned as following:

The new fuzzy center:

$$C_{M+1} = x^k \ , \tag{3.25}$$

$$\sigma_{M+1,1} = \sigma_{M+1,2} = ..., = \sigma_{M+1,N} = 1 , \tag{3.26}$$

$$\mu_{M+1} = \prod_{j=1}^{N} e^{-\frac{(x_j^k - C_{M+1,j})^2}{2\sigma_{M+1,j}^2}} \ . \tag{3.27}$$

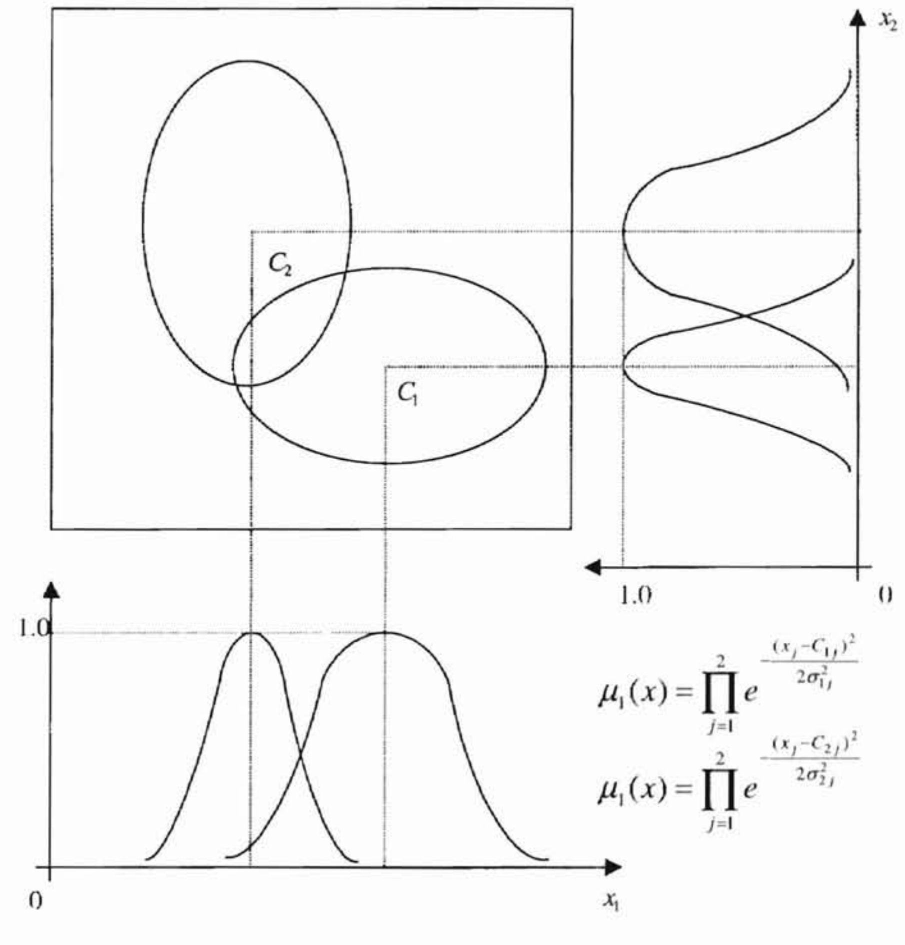
The new sample set S_{M+1}:

$$S_{M+1} = \{x^k\} . \tag{3.28}$$

Also, the new expert network is built up and trained by the sample set S_{M+1}. Figure 3.6 shows two fuzzy clusters overlapping in a two-dimensional input space.

When the observation (x^k, t^k) does not satisfy the two novelty criteria, the Resilient Back-propagation training algorithm is used to adapt the winning MLP expert network parameters as described in Section 3.1, while other losing experts remain unchanged. The fuzzy center is updated by calculating the mean of the winning sample set S_{i^*}:

$$C_{i^*} = \frac{1}{R}\sum_{r=1}^{R} x^r , \tag{3.29}$$



x_2
C_2
C_1
1.0
0
1.0
0
x_1
$\mu_1(x) = \prod_{j=1}^{2} e^{-\frac{(x_j - C_{1j})^2}{2\sigma_{1j}^2}}$
$\mu_1(x) = \prod_{j=1}^{2} e^{-\frac{(x_j - C_{2j})^2}{2\sigma_{2j}^2}}$

Figure 3.6: Two Fuzzy Clusters Structure

where R is the number of the samples in $S_{i^*} = \{x_{i^*}^1, x_{i^*}^2, ..., x_{i^*}^R\}$. The variance of the fuzzy cluster is determined by both the previous variance and the current statistic variance of the updated sample set $S_{i^*} = \{S_{i^*}, x^k\}$. We use a momentum to combine these two factors together:

$$\sigma_{i^*}^2(t) = \gamma\sigma_{i^*}^2(t-1) + (1-\gamma)\frac{1}{R}\sum_{r=1}^{R}(x^r - C_{i^*})^2 . \tag{3.30}$$

When the first several samples are added to the sample set S_{i^*}, the statistical variance will be close to zero. The momentum helps to avoid the risk of zero variance and also keep the statistical characteristics of the variance.

The last step is executed after the whole training samples were trained. The number of samples in each sample set is always showed dramatically different. So, the purpose of the last step is pruning the expert that contains very little training samples and classifying these samples to the other fuzzy cluster by Euclidean distance. Then, the whole experts are updated with the new sample sets.

The overall training step is described as follows.

Growing Fuzzy Clustering Algorithm

Inputs: The training patterns $\{x^k\}$ and the corresponding targets $\{y^k\}$; number of input nodes N ; number of training patterns P ;

Outputs: The experts $\{E_i\}$; the sample sets $\{S_i\}$;

Step 1: determine the architecture of MLP expert based on Vapnik and Chervonenkis theory. [VaCh71] [BaHa89]

Step 2: initialize the first expert:

$C_1 = x^1$;

$\sigma_{11} = \sigma_{12} =,\ldots = \sigma_{1N} = 1$;

```
        $\mu_1(x) = \prod_{j=1}^{N} e^{-\frac{(x_j - C_{1j})^2}{2\sigma_{1j}^2}}$;

        $S_1 = \{x^1\}$;

        Number of experts: $M = 1$;

        train the first MLP expert by sample set $S_1$.

Step 3: for $k = 1$ to $P$ do

            for $i = 1$ to $M$ do

                $\mu_i(x) = \prod_{j=1}^{N} e^{-\frac{(x_j^k - C_{ij})^2}{2\sigma_{ij}^2}}$;

                $a = compet(\mu)$, select the winner expert $E_{i^*}$;

                if $\|x^k - C_{i^*}\| > \varepsilon_1$ and $e^k = y^k - f(x^k) > \varepsilon_2$, do

                    $C_{M+1} = x^k$;

                    $\sigma_{M+1,1} = \sigma_{M+1,2} = ..., = \sigma_{M+1,N} = 1$;

                    $\mu_{M+1} = \prod_{j=1}^{N} e^{-\frac{(x_j - C_{M+1,j})^2}{2\sigma_{M+1,j}^2}}$;

                    $S_{M+1} = \{x^k\}$;

                    train MLP expert $E_{M+1}$ by sample set $S_{M+1}$.

                else, do

                    $S_{i^*} = \{S_{i^*}, x^k\}$;

                    $C_{i^*} = \frac{1}{R}\sum_{r=1}^{R} x^r$; ($R$ is the number of samples in $S_{i^*}$)
```

$$\sigma_{i^*}^2(t) = \gamma\sigma_{i^*}^2(t-1) + (1-\gamma)\frac{1}{R}\sum_{r=1}^{R}(x^r - C_{i^*})^2 ;$$

train MLP expert E_{i^*} by sample set S_{i^*}.

Step 4: Prune the small sample set, class the samples by Euclidean distance, and update the whole experts with the new sample sets.

3.7 Sensor failure mode detection algorithm

The first part of Winner Take All Experts network provided us the estimated critical sensor value when other sensors measurements are used as inputs. To validate a sensor signal, the validation algorithm should trigger an alarm when the sensor signal significantly deviates from the corrected value.

In order to build up the validation gates, the characteristics of both the sensor and the monitored operating environment have to be learned. The validation scheme proposed in this thesis contained both time domain and frequency domain sensor failure detection by using three fuzzy sensor validation gates. So, in learning stage, the statistical information both in time domain and frequency domain has to be learned and recorded by three memories. The first one is used to store the minimum, maximum, and mean value of the sensor signal in training data set. The other two memories are used to record the variation of autocorrelation coefficients r of the residual of the sensor signal, and power spectrum p of the time series. A time window was specified in order to calculate r and p, which are defined as following:

Definition 1: For a series of data $\{x^k : k = 1,...,K\}$, the n th auto-covariance coefficient is defined as:

$$g_n = \sum_{k=n+1}^{K} (x^k - \overline{x})(x^{k-n} - \overline{x}) / K$$

Where $\overline{x}$ is the sample mean:

$$\overline{x} = (\sum_{k=1}^{K} x^k) / K .$$

Then the n th auto-correlation coefficient is

$$r_n = g_n / g_0 .$$

Definition 2: For a series of data, $\{x^k : k = 1,...,K\}$ can be represented by its Fourier series:

$$x^k = (1/K) \sum_{n=0}^{K-1} C_n e^{j2\pi nk/K} ,$$

where C_n is the Fourier coefficients.

Then, Power Density Spectrum Coefficient can be defined as:

$$p_n = |C_n|^2 .$$

After each r and p corresponding to each time window in the learning data set is available, the time series, the auto-correlation coefficients, and the power spectrum density of the sensor signal are learned and analyzed with the eight commonly found sensor failure modes. [Yung92] As shown in the following Figure 3.7, it is worth noting that by inspecting the signatures from different feature spaces, it is easier to identify a good sensor from a faulted one.

In the figure, the first column shows the eight failure modes signal comparing with the nominal state signal in time domain. The other two columns are used to compare the autocorrelation coefficients of the residual, and the power spectrum density for the interested frequency range (for example, in the Westland data set, 3kHz to 10kHz). It is also clear from the figure that using signatures from only one feature space is not sufficient to detect the normal signal form all the other failure modes signals. However, it is easier to detect the faults by combining the comparison results from all three feature spaces.

In order to measure the distances between objects or points in the feature space, a distance measure is used. There are a number of distance metrics that can be used as a tool to measure a similarity between vectors, for example, Euclidean distance, Mahalanobis distance, and Minkowski distance. Without loss of generality, to calculate the difference between the signatures the Manhattan distance is chosen as the distance between two vectors measured along the orthogonal axes.

$$d = |x_1 - y_1| + |x_2 - y_2| + ... + |x_n - y_n|, \tag{3.31}$$

where $x = [x_1, x_2, ..., x_n]$, $y = [y_1, y_2, ..., y_n]$ are the two vectors to be measured.

To investigate the variation of the autocorrelation coefficients r for each time window, the Manhattan distances between each vector are calculated with result d_i, where $i = 1, ..., m$, $m = \frac{n(n-1)}{2}$. n is the number of the time window in the training data set. The minimum, maximum, and mean value of d_i is recorded as the variation characteristics of the autocorrelation coefficients r. The average vector $\bar{r}$ is also recorded for comparison purpose in the detecting stage.

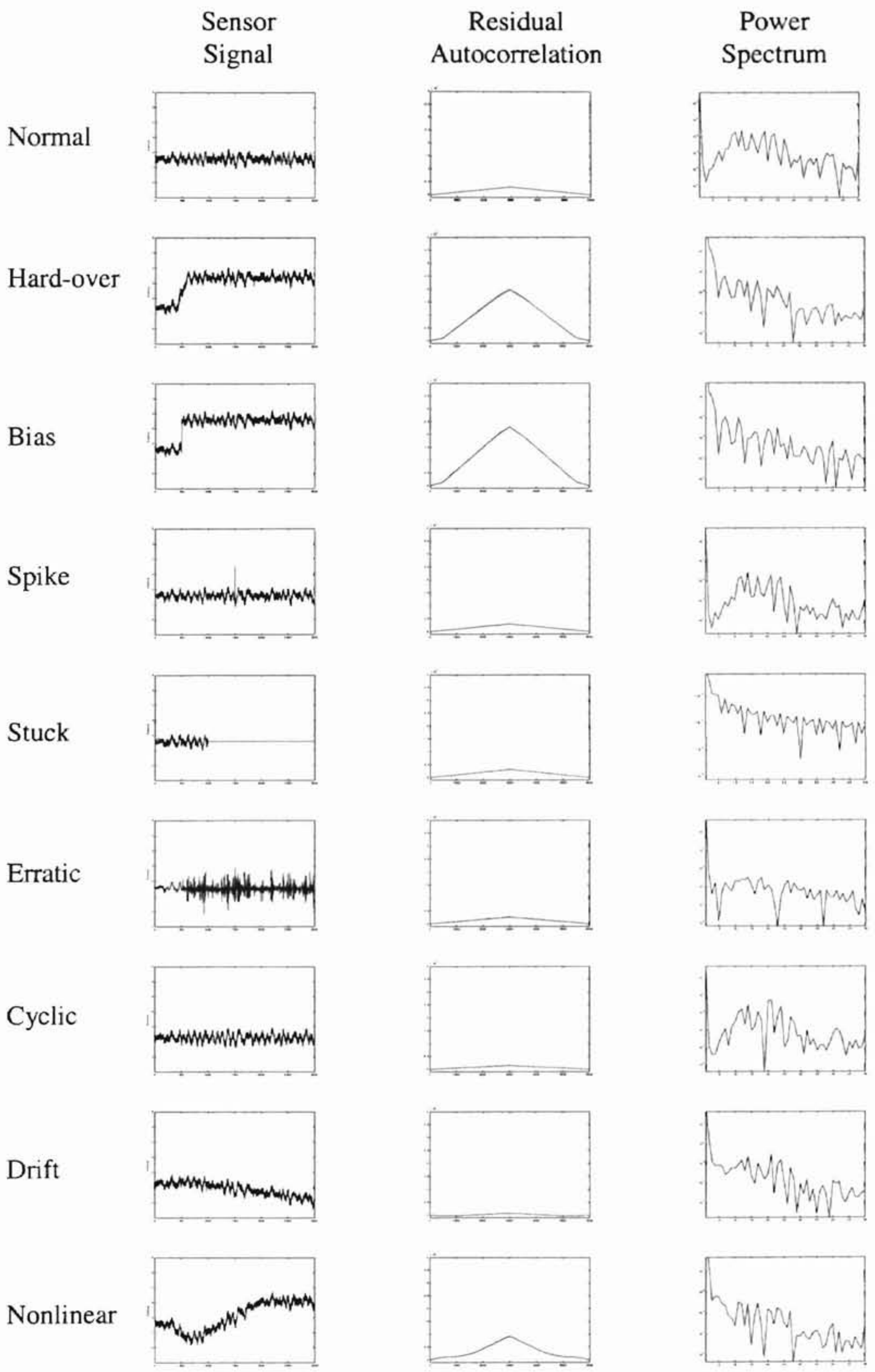


Figure 3.7: Feature Spaces of Sensor Signals

Similar procedure is applied to the power spectrum density p .

Three Validation Gates were established using the information from learning stage. The gates are Sensor Value Validation Gate, Residual Autocorrelation Validation Gate, and Power Spectrum Validation Gate. The basic structure of each gate is based on the dynamic fuzzy validation curve introduced in the second chapter. The mean values from the three memories are set as the highest confidence value for the bell shape validation function. The minimum and the maximum values are set as 10% confidence value points as show in the following figure.

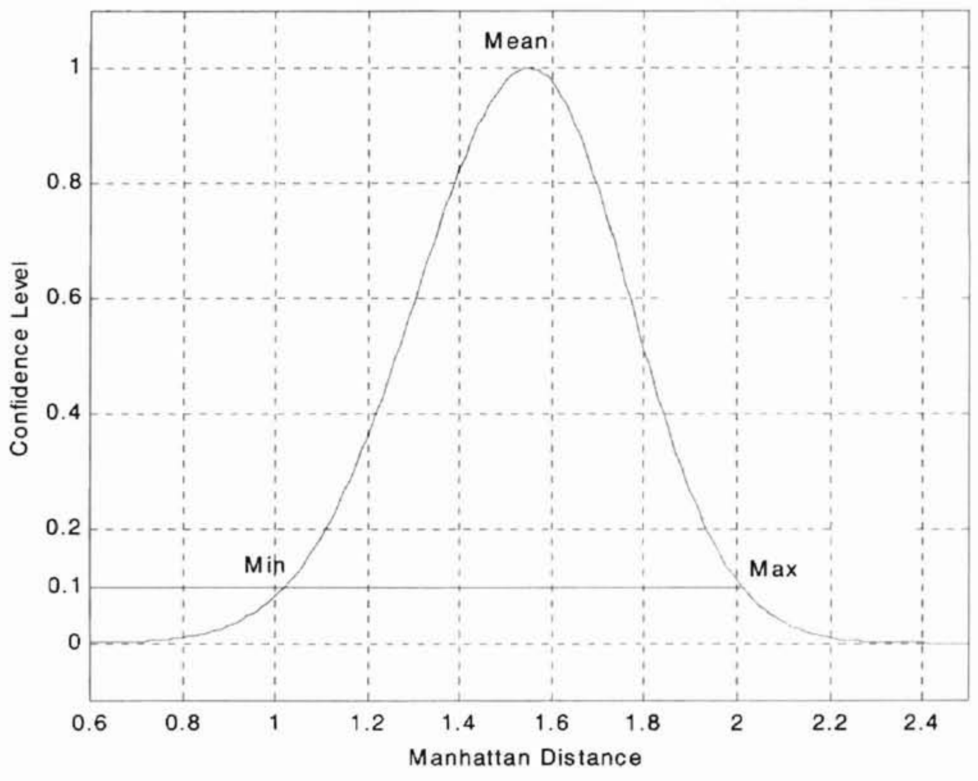


Figure 3.8: A Bell Shape Fuzzy Validation Gate

In detecting stage, the auto-correlation of the residual of the sensor output was calculated every time window, which was compared with the mean autocorrelation coefficients $\bar{r}$ in Manhattan distance by the corresponding validation gate. The output will be the sensor health confidence level in autocorrelation feature domain. Similar approach is applied for the power spectrum density and each sensor output datum.

Finally, the three outputs from the three gates are fused by fuzzy intersection mentioned in Section 3.3. The decision level fusion output is a final confidence level of the interested sensor health.

Fuzzy Sensor Validation Algorithm

Input: The signal from real sensor; the estimation value from winner take all experts network; the time window size; the length of FFT.

Outputs: the confidence level of the sensor.

```
Step 1: Filter the real sensor signal $y$ and the estimation value $\hat{y}$ from Winner Take All
        Experts Network by time window;
Step 2: Transfer the two time series to frequency domain by FFT with the given length;
Step 3: Calculate the Power Spectrum Density $p_1$ and $p_2$ of the two time series;
Step 4: Calculate the residual of the $r_1$ and $r_2$ of the two time series;
Step 5: Compare $p_1$ and $p_2$ by Manhattan distance with result $d_p$;
Step 6: Compare $r_1$ and $r_2$ by Manhattan distance with result $d_r$;
Step 7: Calculate the confidence level of the sensor by the three validation gates;
```

Step 8: Decision level fusion by intersecting the three fuzzy sets.

3.8 Summary of the WTAE algorithm

The WTAE algorithm divides a complicated problem into a series of sub-problems. It uses a growing fuzzy membership clustering method to divide the input space into overlapping regions on which 'experts' act, and the Gaussian membership function localizes the expert's influence into a region around the cluster center. A simplification of the criterion leads to two joint criteria on the distance of the present pattern and the existing unit centers in the input space, and on the approximation error of the network for the given observation to be satisfied together. The last step, comparisons between the real sensor value and estimation signals are based on three validation gates. The first validation gate, sensor value validation gate, compares the tracking output directly with the real sensor data. In the second validation gate, the residual of the two series is investigated by the autocorrelation coefficient. The power spectrum density of the two signals is compared in the last validation gate. A decision level fusion of the three validation gates is accomplished by the intersection of the three fuzzy sets. The fusion result is considered to be more reliable and robust than using only one validation gate alone.

CHAPTER IV

SIMULATION RESULTS

Software simulation is used in our experiments to demonstrate the expected performance of the WTAE network. The simulation programs were coded under the MATLAB software environment (version 5.2). A Pentium 233MMX PC hosted the simulation programs. Two data sets were used for training and testing the sensor validation system in our studies. The first benchmark data set was generated from a Spectra Quest (SQ) Machinery Fault Simulator. The second data set was a time-series vibration data set, commonly known as Westland vibration data.

4.1 SQ Machinery Fault Simulator data set

This data set consists of vibration data recorded from SQ Machinery Fault Simulator, which is shown in Figure 4.1. The instrument is constructed with special kinds of bearings, rotors with split collar ends, and a split bracket bearing housing. The simulator offers a wide range of predictive maintenance and the signatures of various bearing faults.

Because the purpose of this thesis is sensor fault detection, not the machinery fault identification, the signal used in the simulation is the nominal system state signal. Figure 4.2 shows one sample time series from the accelerometer of the simulator.

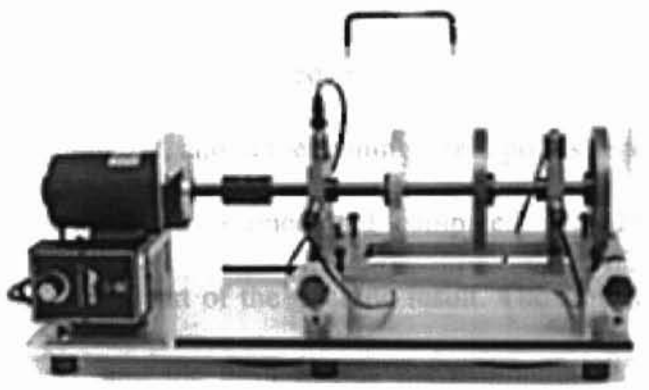


Figure 4.1: SQ Machinery Fault Simulator

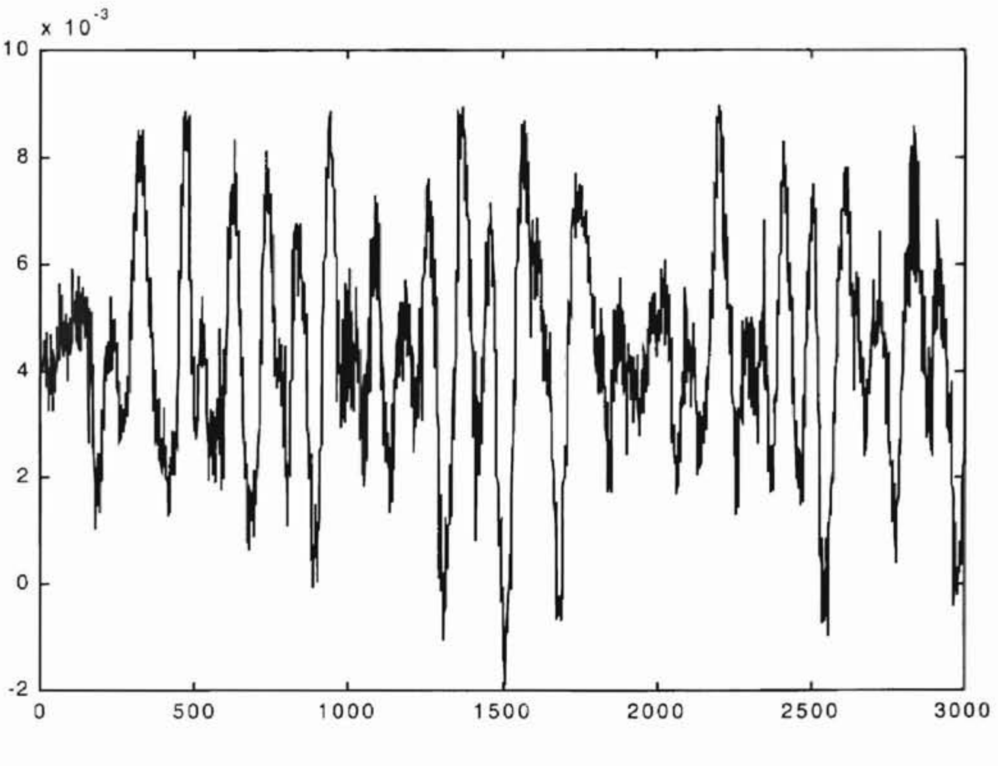


Figure 4.2: A Time Series of SQ Machinery Simulator Sensor Signal

Here, we use another three accelerometer sensors in proximity, which have the similar high frequency waveform, to approximate the sensor measurements of interested. The relationship of the sensors is obviously nonlinear, and the modeling work is very complex.

The sample data set was separated into two parts. The first 3,000 observation pairs are used as the training set, and the remaining 3,000 points serve as the testing set.

The WTAE network was trained and compared with MLP and RBF based estimators. Figure 4.3 shows part of the tracking result. The solid line is the real output from sensor (the target), and the dash line is the estimation value from the WTAE network.

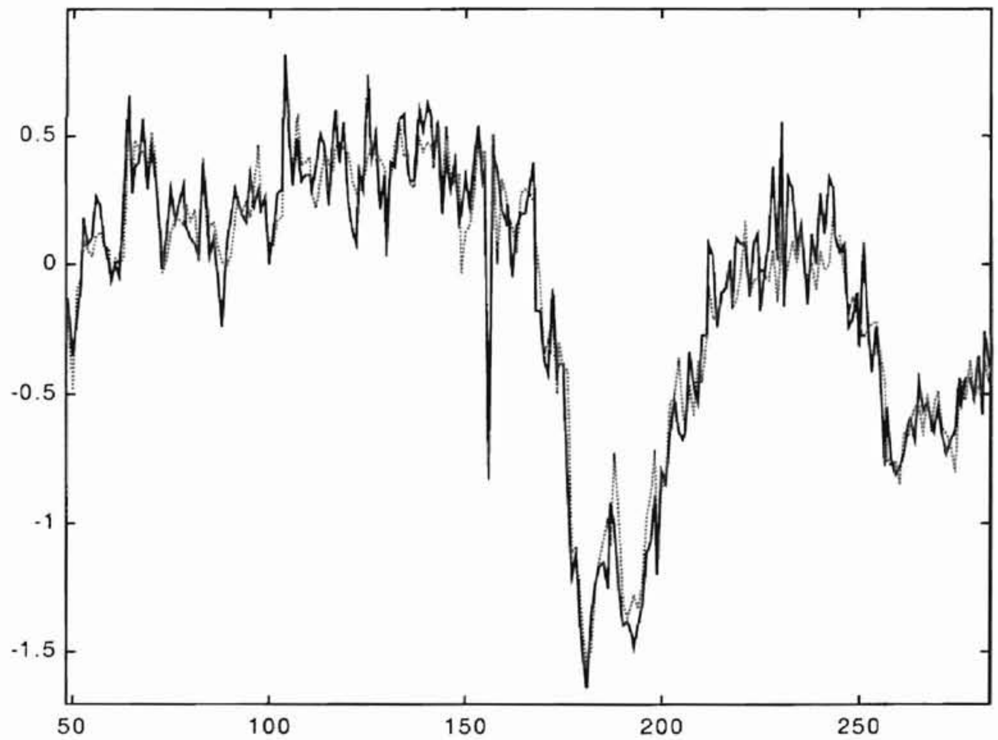


Figure 4.3: Outputs of WTAE and Targets of Training Set

The performance function for the experts is mean square error (MSE) function, the average squared error between the network outputs and the target output. A ten hidden nodes MLP structure was heuristically chosen to be the local expert with Resilient Back-propagation training algorithm. After the WTAE network has been trained for totally 41 minutes, the MSE of the training set equals to 0.0234. The resulted network incorporates with 73 experts, with a pruning criterion 10 samples per sample set. Part of the testing result is shown in Figure 4.4:

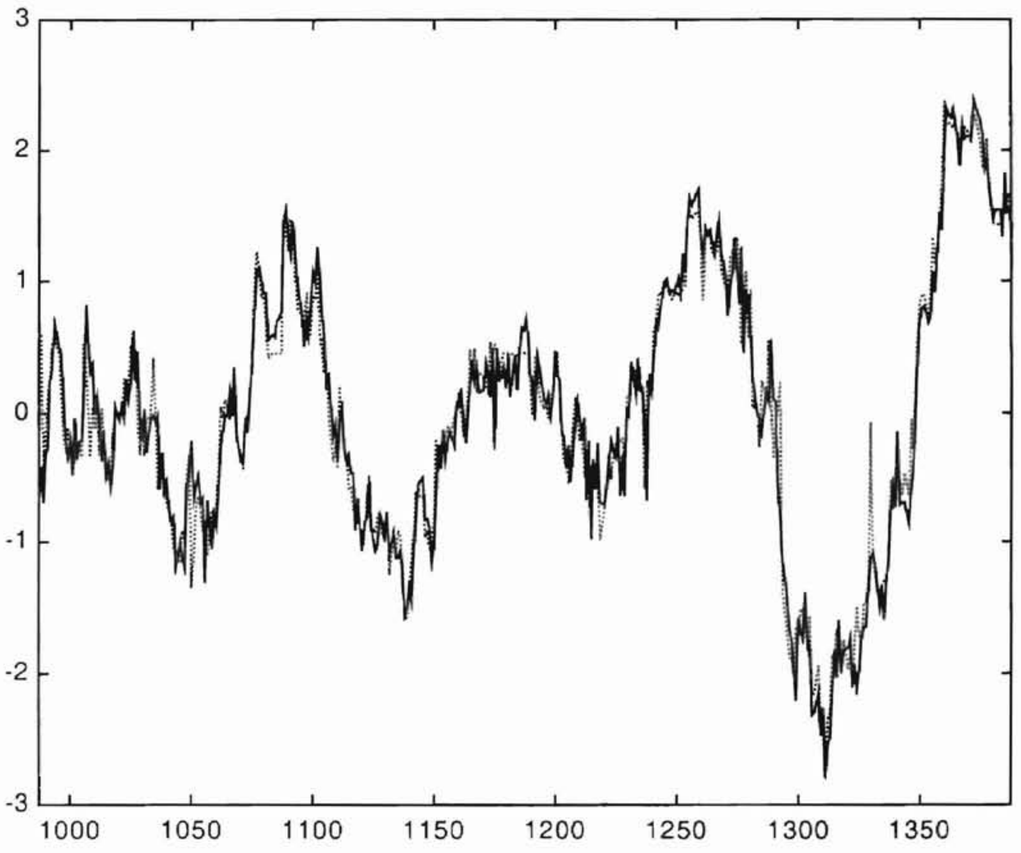


Figure 4.4: Outputs of WTAE and Targets of Testing Set

The mean square error of the 3,000 testing samples using WTAE network is 0.1631.

The same training set was also applied for training MLP network. The total number of nodes of MLP is determined by VCdim theory [VaCh71] [BaHa89], which is estimated to be 300. After training several pre-selected structures, the structure with the best result is a MLP with a 150-node hidden layer. The network was trained by Resilient Back-propagation algorithm as well. After 41 minutes, the MSE is 0.0554. Also, the network were tested by the same 3,000 samples as the WTAE network did. The resulted MSE is 5.9529.

The third estimation method used for comparison is RBF network. Here, we used the k-Means Clustering Algorithm to train the first layer and the Moore-Penrose algorithm for the second layer as described in Chapter 3. Because of the high computational cost involving matrix inversion, the RBF network can only be trained by 775 samples in 41 minutes. The resulted MSE for the training set is 0.0550. The resulted number of first layer clusters is 339 because of the complexity of the input space. The next 775 samples were tested with a MSE 0.0787.

From the above simulation results, we draw the following conclusions:

1. WTAE network is a model free algorithm: Just like any other neural network based method, WTAE can also uniformly approximate any continuous function without knowledge of system model, as long as the number of experts units is sufficient. This is because the basic unit of the network is a small single hidden layer MLP network, which realize the nonlinearity of the system locally. It is a great advantage of neural network based algorithm over the traditional knowledge based tracking system.

2. WTAE network have outperformed other neural network based system in complex problems, Based on the 'divide and conquer', the model employs a growing fuzzy clustering method naturally divide the input space into overlapping regions on which 'experts' act. In this way, a complicated problem was divided into a series of simpler sub-problems and assigned a set of function approximators to each sub-problem.
3. The growing fuzzy clustering algorithm is able to build clusters of both linear and nonlinear separable decision boundaries: In most of sensor value tracking problems, the input sample sets are always nonlinear separable. So, the ability of building the decision boundaries to separate both linear and nonlinear separable classes is necessary.
4. The growing fuzzy clustering algorithm is able to make decision boundaries of overlapping classes: The overlapping is another feature of the input sample sets in sensor tracking problems. The algorithm we used has overcome the overlapping problems by using Gaussian fuzzy membership functions in each dimension of input space.

A comparison performance among the three algorithms were listed in Table 4.1:

Table 4.1: A Comparison Performance Among Estimators

41 Mins Training	WTAE network	MLP network	RBF network
Architecture	73 of 10-hidden nodes MLP experts	150 hidden nodes	339 clusters in the first layer
Training Algorithm	Growing fuzzy Resilient-BP	Resilient-BP	k-means clustering Moore-Penrose
Training samples	3,000	3,000	775
MSE of training	0.0234	0.0554	0.0550
Testing samples	3,000	3,000	775
MSE of testing	0.1631	5.9529	0.7874

A time window was defined with length of 300 points. Both the real sensor time series and the tracking time series were transferred to frequency domain by FFT with given length 256. Both the power spectrum density and the autocorrelation of the residual were calculated. The necessary information for establishing the three fuzzy sensor validation gates was recorded. Based on the collected information the following validation gates were built:

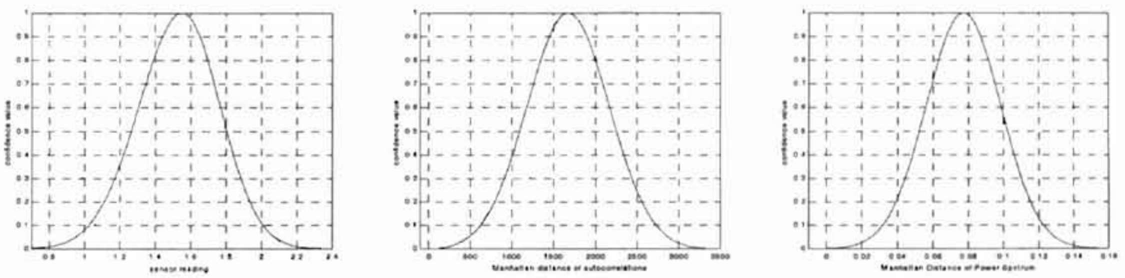


Figure 4.5: Three Fuzzy Sensor Validation Gates

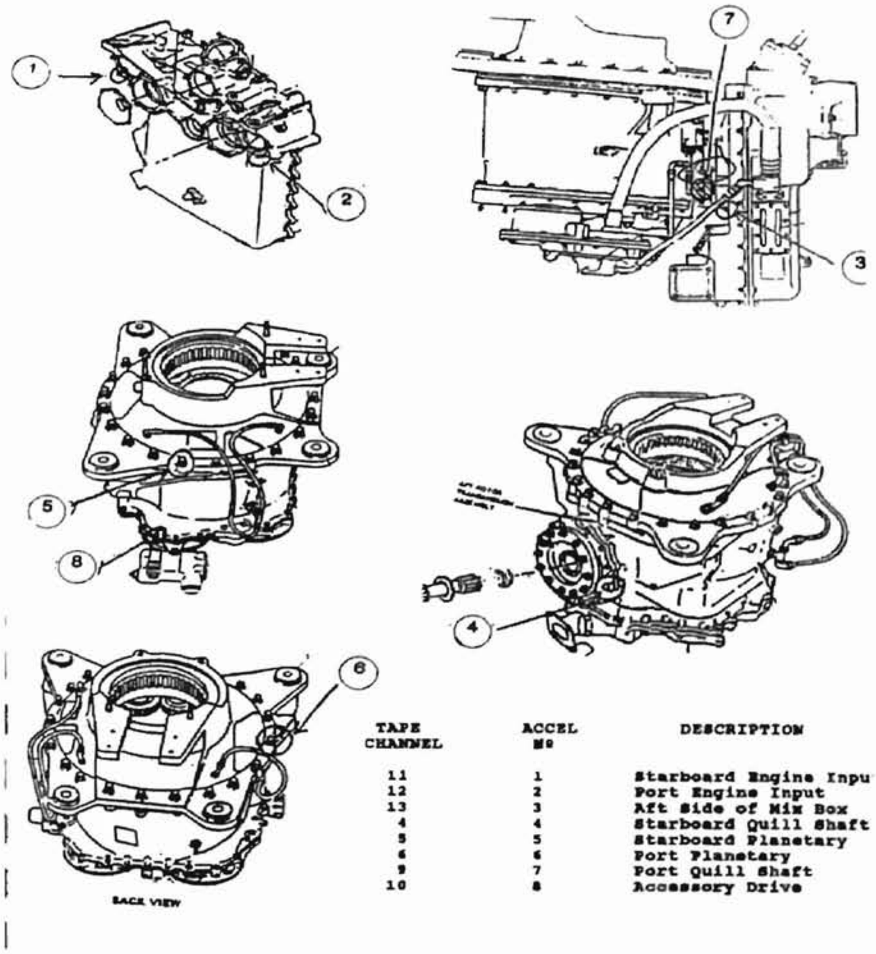
The results from the three validation gates and the final confidence level from fuzzy intersection are shown in Table 4.2. Compared with the traditional time domain indicators, WTAE network is more reliable and robust. When the amplitude of noise is not big enough to trigger the noise indicator, the traditional indicators could not detect several failure modes, especially 'Cyclic', which means in the same condition the traditional indicators could only detect less than 87.5% failure states. In addition, the traditional indicators need sufficient knowledge of characteristics of both the sensor and the environment, which is unnecessary for the proposed WTAE network.

Table 4.2: Comparison Results of Three Validation Gates and Fuzzy Intersection Output

Sensor States	Sensor Value Validation Gate	Residual Autocorrelation Validation Gate		Power Spectrum Validation Gate		Fuzzy Intersection
		Manhattan Distance	Confidence Level	Manhattan Distance	Confidence Level	
Normal	0.9046	1.6723e+003	0.9832	0.0771	0.9447	0.9046
Hard-over	2.8921e-014	6.1907e+005	1.0613e-008	2.7423	1.0854e-024	1.0854e-024
Bias	7.1865e-016	7.2108e+005	1.0613e-008	3.1193	1.0854e-024	1.0854e-024
Spike	7.1858e-016	1.6460e+003	0.9989	0.0822	0.9721	7.1858e-016
Stuck	0.8460	5.6381e+003	1.0591e-008	1.5572	1.0854e-024	1.0854e-024
Erratic	0.9268	1.5550e+004	1.0613e-008	0.4496	1.0854e-024	1.0854e-024
Cyclic	0.4501	4.4328e+004	1.0613e-008	0.4698	1.0854e-024	1.0854e-024
Drift	1.2242e-006	6.1678e+004	1.0613e-008	0.8432	1.0854e-024	1.0854e-024
Nonlinear	5.2260e-014	3.1871e+004	1.0613e-008	1.1034	1.0854e-024	1.0854e-024
Correction Rate	66.67%	88.89%		88.89%		100%

4.2 The Westland vibration data set

The Westland data set was acquired using an array of eight accelerometers fixed in specific locations on a set of faulted and unfaulted aft main power transmission of a U.S. Navy CH-46 helicopter. These accelerometer-equipped transmissions were mounted on a laboratory-based "test rig" and run at a sampling rate of 103,116.08 Hz. The sensor validation experiment data set was sampled at no-fault condition at one of the several torque load levels (i.e., 100%).



TAPE CHANNEL	ACCEL Nº	DESCRIPTION
11	1	Starboard Engine Inpu
12	2	Port Engine Input
13	3	Aft Side of Mix Box
4	4	Starboard Quill Shaft
5	5	Starboard Planetary
6	6	Port Planetary
9	7	Port Quill Shaft
10	8	Accessory Drive

Figure 4.6: Accelerometer Positions on the AFT Transmission

As shown in Figure 4.6, sensor 4, 5, 6, and 8 are located within a neighborhood. Assuming they are related to each other, we choose sensor 8 as the target of our estimator, and the other three as the input signals. Figure 4.7 shows one sample time series from the Westland Data Archive.

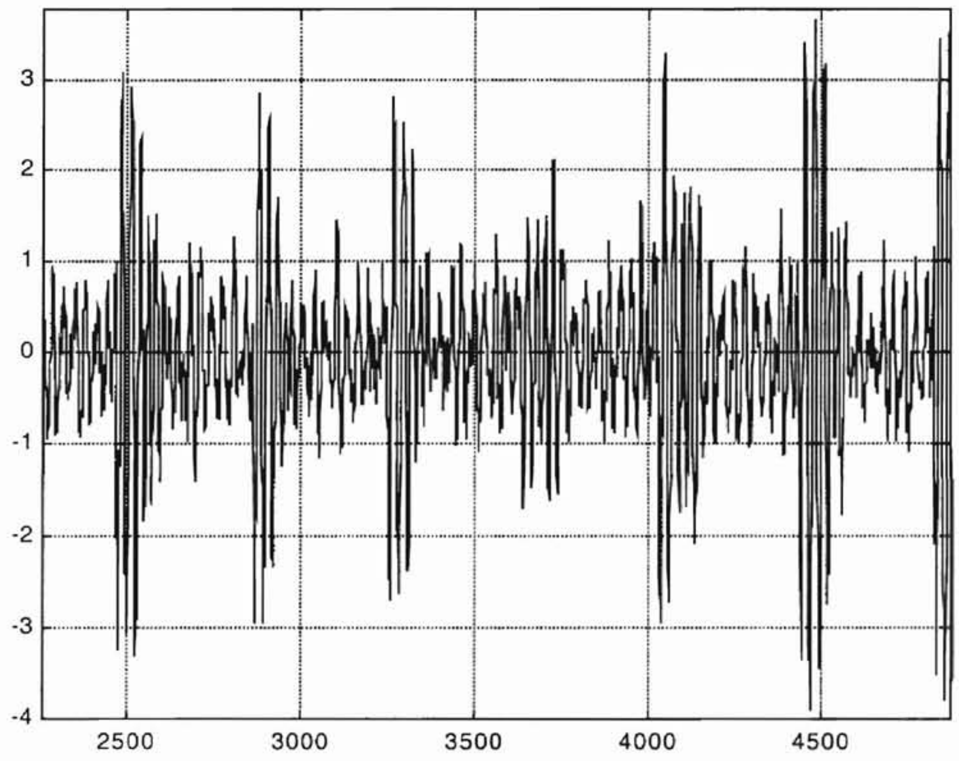


Figure 4.7: A Time Series of Westland Data Set

Because of the complexity of the data set, we included 3,000 samples both in the training set and testing set. The estimation results were listed in Table 4.3:

Table 4.3: A Comparison Performance Among Estimators (2)

63 Mins Training	WTAE network	MLP network	RBF network
Architecture	50 of 15-hidden nodes MLP experts	150 hidden nodes	385 clusters in the first layer
Training Algorithm	Growing fuzzy Resilient BP	Resilient BP	k -means clustering Moore-Penrose
Training samples	3,000	3,000	813
MSE of training	0.3985	0.7756	42.5571
Testing samples	3,000	3,000	813
MSE of testing	2.1402	45.9567	42.3122

Although we added more hidden neurons to the experts than the first application, the MSE of both training and testing sets are still big. The other two networks also had the similar results as WTAE network. The possible reason is that the sensors used are probably not strongly correlated. That means the assumption we made before building up the estimator is probably invalid. Further study needs to be invested.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions of the research

An architecture for estimating sensor measurements and detecting sensor failure has been developed in this thesis. The method allows us to estimate a critical sensor data when other neighboring sensors measurements are used as inputs. Three fuzzy sensor validation gates based on the information from both time domain and frequency domain were used to detect the sensor failure.

The network is a synergetic combination of fuzzy logic and neural network. It employs the fast parallel computation and learning capability of neural network. In addition, fuzzy set theory adds the ability to represent and manipulate imprecise information.

The WTAE network consists of two main layers: Fuzzy membership clustering layer and MLP experts layer. The cluster layer employs the Gaussian radial basis function as a fuzzy membership function. The general idea is to divide a complicated problem into a series of simpler sub-problems and assign a function approximator to each sub-problem. A growing fuzzy membership clustering method was used to divide the input space into overlapping regions on which 'experts' act.

After the WTAE network was fully trained in sensor nominal state, the estimation result was compared with real sensor outputs by three fuzzy validation gates, which were built up based on the information collected in the learning stage. The auto-correlation of the residual and the power spectrum of the time series are analyzed by Manhattan distance. The results from the three validation gates were combined together by fuzzy intersection logic. This decision level fusion finishes the whole sensor failure detection procedure providing the final confidence level of the interested sensor health.

Two benchmark data set: the SQ Machinery Fault Simulator data set and the Westland vibration data set were used in simulation studies to demonstrate the performance of the WTAE network. Comparisons between the WTAE network and the other two neural networks estimators were made. The results show that, in terms of estimation performance (MSE), the WTAE is competitive with or even superior to the MLP network and RBF network.

Furthermore, the fuzzy sensor validation gates algorithm was used to investigate the eight sensor failure modes. The results from the simulation studies have shown that the proposed validation algorithm is efficient for detection all of the eight faults as long as the basic assumption, that the neighboring sensors do exist an analytical relationship, is valid.

5.2 Recommendation of possible future work

The future research could concentrate on distinguishing the sensor failure mode from the system failure mode. A possible approach was described below.

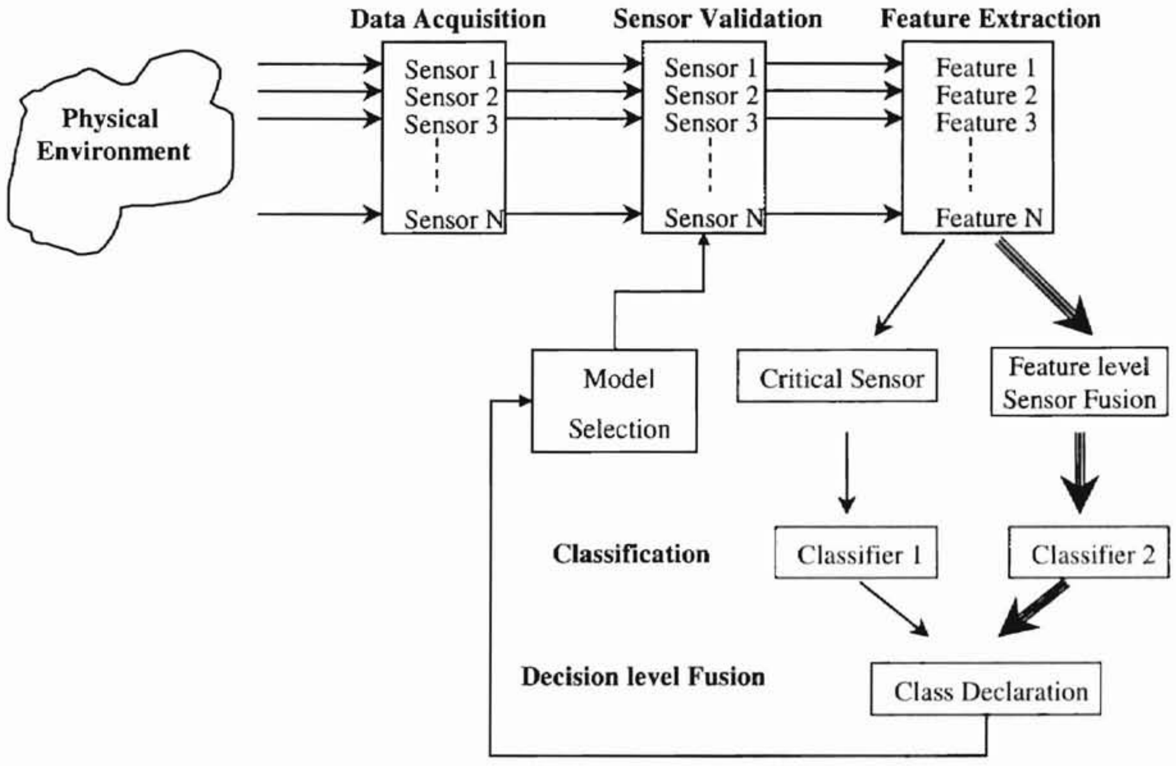


Figure 5.1 Sensor Validation and Fusion Structure

Figure 5.1 shows a simple architecture of machine health monitoring system. The characteristics of physical plant are monitored by sensors. From the sensors, digital signal processing and feature extraction are used to preprocess data, as well as the sensor validation part. The data was validated, and at the same time, the dimension of data was reduced in order to obtain patterns containing enough information to discriminate in a lower dimension. Next, pattern classification classifies and identifies the types of fault conditions in two different channels. One is the critical sensor channel, which plays a dominated role in system fault identification. The other one is the feature level fusion channel, which combined the information from all sensors. The results then are integrated by a decision level sensor data fusion algorithm. [Luka89] [FiMi90]

The sensor validation network will have several estimation models corresponding to each system failure modes. They will be trained off-line under these anticipated failure modes. After the training, the system will operate in the following: when the sensor validation alarms, the signals are put forward through the system to the output. The class declaration part will give out a declaration of which state the system is in. This result not only depends on the critical sensor, but also from the other related sensors monitoring the same operating system. Therefore, even though we have a fault critical sensor, we still have a correct system state result with high possibility. Then, based on this result, the model selector will decide which WTAE model will be used for current sensor validation, and the critical sensor will be validated again. If the validation result repeats the same alarm (trending), we can draw a conclusion that the sensor is in failure mode and the operator will be instructed to take necessary actions.

REFERENCE

1. [Ande72] Anderson, J., "A simple neural network generating an interactive memory." Mathematical Biosciences, 14, 1972, pp. 197-220.

2. [BaHa89] Baum, E. and Haussler, D., "What size net gives valid generalization?" Neural Computation, 1, 1989, pp. 151-160.

3. [BlRi88] Blum, A. and Rivest, R., "Training a 3-node neural network is NP-complete." In Proceedings of the Computational Learning Theory Conference, 1988, pp. 9-18.

4. [Brow92] Brownell, Thomas, "Neural networks for sensor management and diagnostics." In Proceedings of the IEEE International Conference on Neural Networks, 1992, pp. 923-929.

5. [ChPu88] Chandrasekaran, B. and Punch, William, "Hierarchical classification: its usefulness for diagnosis and sensor validation." IEEE Journal on selected areas in communications, 6(5), 1999, pp. 884 –891.

6. [Clar95] Clarke, D., "Sensor, Actuator, and Loop Validation." IEEE Control System Magazine, 15, 1995, pp. 39-45.

7. [ErUp90] Eryurek, E. and Upadhyaya, B., "Sensor validation for power plants using adaptive back-propagation neural network." IEEE Transaction on Nuclear Science, 37(2), 1990, pp. 1040 –1047.

8. [EsDi98] Essawy, M., Diwakar, S., and Sabatto, S., "Fault diagnosis of helicopter gearboxes using neuro-fuzzy techniques." In Proceedings of the 52nd Meeting of the Society for Machinery Failure Prevention Technology, 1998, pp. 381 –385.

9. [FePu98] Feldkamp, Lee and Puskorius, Gintaras, "A signal processing framework based on dynamic neural networks with application to problems in adaptation, filtering, and classification." In Proceeding of the IEEE, Vol. 86(11), 1998, pp. 2259 –2277.

10. [FiMi90] Fincher, D. and Mix, Dwight, "Multi-sensor data fusion using neural networks." In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 1990, pp. 835 –838.

11. [Goeb96] Goebel, Kai, "Management of uncertainty in sensor validation, sensor fusion, and diagnosis of mechanical system using soft computing techniques," D. Phil. Thesis, Mechanical Engineering, University of California at Berkeley, 1996.

12. [GoRi94] Gonzalez, G. and Richard, J., "Issues in soft sensor applications in industrial plants." Industrial Electronics, Symposium Proceedings, 1994, pp. 380-385.

13. [GuNu91] Guo, T. and Nurre, J., "Sensor failure detection and recovery by neural networks." IJCNN International Joint Conference, 1, 1991, pp. 221 -226.

14. [Habt98] Habtom, Ressom, "Soft-sensing using recurrent neural networks." In Proceedings of the IEEE/ISIC/CIRA/ISAS Joint Conference, 1998, pp. 342 –347.

15. [HaLi98] Habtom, Ressom and Litz, Lothar., "Virtual sensors based on recurrent neural networks and the extended Kalman filter." In Proceedings of the IEEE, International Conference on Control Applications, 1, 1998, pp. 163 –167.

16. [Himm95] Himmelblau, David, "On-line sensor validation of single sensors using artificial neural networks." In Proceedings of the American Control Conference, 1995, pp. 456 –468.

17. [Horn89] Hornik,K., "Multilayer feedforward networks are universal approximators." Neural Networks, 2, 1989, pp. 359-366.

18. [JoJa94] Jordan, Michael and Jacob, Robert, "Hierarchical mixture of experts and the EM algorithm," Neural Computation, 6, 1994, pp.181-214.

19. [KaNi93] Kadirkamanathan, Visakan and Niranjan, Mahesan, "A function estimation approach to sequential learning with neural networks." Neural Computation, 5, 1993, pp. 954-975.

20. [Lee94] Lee, S., "Sensor value validation based on systematic exploration of the sensor redundancy for fault diagnosis KBS." IEEE Transaction on Systems, Man, and Cybernetics, 24(4), 1994, pp. 594 –605.

21. [LuKa89] Luo, Ren and Kay, Michael, "Multisensor integration and fusion in intelligent system." IEEE Transaction on Systems, Man, and Cybernetics, 19(5), 1989, pp. 901 –931.

22. [Manu95] Manus, Henry, "Sensor validation and fieldbus." Computing & Control Engineering Journal, 1995, pp. 263 –269.

23. [NaWi98] Napolitano, Marcello and Windon, Dale, "Kalman filters and neural network schemes for sensor validation in flight control system." IEEE Transactions on Control Systems Technology, 6(5), 1998, pp. 596 –611.

24. [NaSi98] Napolitano, Marcello and Silverstri, Giovanni, "Sensor validation using hardware-based on-line learning neural networks." IEEE Transactions on Aerospace and Electronics Systems, 34(2), 1998, pp. 456 –468.

25. [Park85] Parker, D., "Learning-logic: casting the cortex of the human brain in silicon," Technical Report TR-47, Center for Computational Research in Economics and Management Science, MIT, 1985.

26. [PaSa91] Park, J. and Sandberg, I., "Universal approximation using radial-basis-function networks." Neural Computation, 3, 1991, pp. 246-257.

27. [RiBr93] Riedmiller, M. and Braun, H., "A direct adaptive method for faster back-propagation learning: the RPROP algorithm." In Proceedings of the IEEE International Conference on Neural Networks, 1993, pp. 170 –175.

28. [RuHi86] Rumelhart, D., Hinton, G. and Williams, R., "Learning representations by back-propagating errors," Nature, 323, 1986, pp. 533-536.

29. [SmGa91] Smith, C. and Erickson, Gary, "Multisensor data fusion: concepts and principles." Communications, Computers and Signal Processing, IEEE, 1, 1991, pp. 235 –237.

30. [VaCh71] Vapnik, V. and Chervonenkis, A., "On the uniform convergence of relative frequencies of events to their probabilities." Theory of Probability and its Applications, 16(2), 1971, pp. 264-280.

31. [WaLo96] Wang, Exuding and Lou, Wrongful, "Designing a soft sensor for a distillation column with the fuzzy distributed radial basis function neural network." In Proceeding of the 35th Conference on Design and Control, 2, 1996, pp. 1714 –1719.

32. [WaMe92] Wang, L. and Mendel, J., "Back-propagation fuzzy systems as nonlinear dynamic system identifiers." In Proceedings of the IEEE International Conference on Neural Networks, Fuzzy Systems, 1992, Pp1409-1418.

33. [Werb74] Werbos, P., "Beyond regression: new tools for prediction and analysis in the behavioral sciences," D. Phil. Thesis, Harvard University, 1974.

34. [WhDh94] Wheeler, Kevin and Dhawan, Atam, "SSME parameter estimation using radial basis function neural networks." IEEE World Congress on Computational Intelligence, Neural Networks, 5, 1994, pp. 3352 –3357.

35. [XuOj93] Xu, Lei and Oja, Erkki, "Rival penalized competitive learning for clustering analysis, RBF net, and curve detection." IEEE Transaction on Neural Networks. 4(4), 1993, pp. 636 –649.

36. [YaCh97] Yang, Yingxu and Chai, Tianyou, "Soft sensing based on artificial neural network." In Proceedings of the American Control Conference, 1, 1997, pp. 674 – 678.

37. [Yung92] Yung, S., "Signal processing in local sensor validation." D.Phil. thesis, Department of Engineering Science, University of Oxford, 1992.

38. [Zade65] Zadeh, L., "Fuzzy sets." Information and Control, 8, 1965, pp. 338-353.

39. [ZhKa94] Zhang, D. and Kamel, M., "Fuzzy clustering neural network (FCNN): competitive learning and parallel architecture." Journal of Intelligent and Fuzzy Systems, 2, 1994, pp. 289-298.

VITA

Wei Feng

Candidate for the Degree of

Master of Science

Thesis: WINNER TAKE ALL EXPERTS NETWORKS FOR SENSOR VALIDATION

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Beijing, China, On January 10, 1974, the son of Jing-Pei Feng and Bao-Zhu Zhang.

Education: Received a Bachelor of Science in Electrical Engineering degree from Xi'an JiaoTong University, Xi'an, China in May 1996. Completed the requirements for the Master of Science degree with a major in Electrical Engineering at Oklahoma State University in May, 2000.

Experience: Employed as an electrical engineer, 1996 to 1998 by IQ Management UK limited, China distributor's office in Beijing; employed by Oklahoma State University, School of Electrical and Computer Engineering as a research assistant and teaching assistant, 1998 to 2000.