Applied Intelligence 9, 231–243 (1998)

A Fuzzy Diagnostic Model and Its Application in Automotive Engineering Diagnosis

YI LU AND TIE QI CHEN
Department of Electrical and Computer Engineering, The University of Michigan-Dearborn, Dearborn Michigan 48128-1491, U.S.A.
yilu@umich.edu

BRENNAN HAMILTON
Diagnostic Systems Department, Powertrain Division, Ford Motor Company, Dearborn, MI 48128

Abstract. This paper describes a fuzzy diagnostic model that contains a fast fuzzy rule generation algorithm and a priority rule based inference engine. The fuzzy diagnostic model has been implemented in a fuzzy diagnostic system for the End-of-Line test at automobile assembly plants and the implemented system has been tested extensively and its performance is presented.

Keywords: fuzzy logic, machine learning, fault diagnosis

1. Introduction

Fault diagnosis has been a classic engineering problem. The techniques for diagnosing faults in dynamic systems range from expert systems to statistic models [1, 2, 3, 5]. In spite of these efforts, automotive engineering diagnosis is continuously being considered as one of the most challenging problems in AI, because the electronic control components (ECC) in modern vehicles are more like a black box; it is difficult for mechanics to diagnose faults accurately unless they are thoroughly familiar with the system specifications and functions [6, 7]. As a result, it is extremely difficult to develop a complete diagnostic model that can fully answer all the questions in terms of classifying faults. Filljov et al. described an expert system for engine diagnosis [3]. The expert system attempted to evaluate the condition of an automobile engine and the electronic system that controls the engine, and then forecast the behavior of the engine and the electronic system. The knowledge rules were generated based on expert knowledge, which is qualitative and incomplete. Zheng et al. described a knowledge-based diagnosis system for automobile engine [7]. The system has a hierarchical structure of diagnosis principles. According to this principle, a complex diagnostic task was divided into several simple tasks to be solved step-by-step. We found these systems are not capable of learning diagnostic knowledge from training data. The theory of fuzzy logic is aimed at the development of a set of concepts and techniques for dealing with sources of uncertainty or imprecision and incomplete. Fuzzy systems have been successful in many applications including control theory when gradual adjustments are necessary, control systems, business and even the stock exchange [9–16]. The nature of fuzzy rules and the relationship between fuzzy sets of differing shapes provides a powerful capability for incrementally modeling a system whose complexity makes traditional expert system, mathematical, and statistical approaches very difficult. The importance of automatically generating rules for intelligent systems has been well recognized in the AI community. Typical approaches include neural network techniques [8, 10, 17, 18], inductive learning algorithms or pseudo-Boolean logic simplification methods [19], and fuzzy c-means clustering method [20]. The algorithm

described in this paper is different from those existing approaches, are extremely effective and efficient. Specifically, the algorithm attempts to automatically generate a compact and optimal set of fuzzy rules.

The algorithm has been implemented in a fuzzy automotive diagnostic system used in the End-of-Line test in automobile assembly plants. We have tested the implemented system on large sets of data of different vehicle models acquired directly from various test sites of Ford assembly plants, and the performance of the system is presented.

2. Fuzzy Modeling of Automotive Engineering diagnosis

As automotive electronic control systems have become more advanced and sophisticated in the recent years, malfunction phenomena have also become increasingly more complicated. It has been well recognized in the automotive industry that effective vehicle diagnostic systems will play a key role in the competitive market. The major US automotive companies have launched an End-of-Line test system at every North American assembly plant. In order to accomplish this task, a test is build to acquire and analyze Electronic Engine Controller (EEC) data while the vehicle is dynamically tested. During the test, operators drive the vehicles through a preset profile and the vehicles are either passed or failed according to the data collected during the test. The decision is made based on two information sources, the EEC on-board test and off-board test performed by the vehicle test system on EEC generated data. We focus on solving the second problem, namely, off-line test. The existing techniques used in assembly plants for the off-line test are mainly on a trial-and-error basis and largely dependent on individual engineers' experience. There is a strong need for developing an intelligent system to automatically detect EEC failures. The nature of this application requires an intelligent system to provide prompt and reliable diagnosis.

This application has the following characteristics:

- Knowledge of most vehicle diagnostic problems is incomplete and vague due to the complexity of the modern vehicles. One reason that causes the incompleteness is that we often do not have a complete set of parameters necessary to fully describe a faulty behavior or component. In general, engineers can use scientific and expert knowledge to define a list of parameters associated with a particular type of fault. However, the actual data acquired at the test site is often limited due to physical limitation. Therefore, some parameters may be inaccurately substituted into the model to serve as surrogates for the unavailable parameters, and other parameters may not be available at all. This uncertainty nature of the problem leads us to seek a solution in fuzzy diagnostic models.
- Different vehicle models have different engineering features, e.g., Ford ThunderBird is very different from Ford Town Car. Engineering aspects of the same model can change year to year. For example, ThunderBird 96 may be different from ThunderBird 97 in certain aspects. We must need an automatic mechanism to learn automotive diagnostic knowledge so the system can quickly adapt to different vehicle models for test.

Engineering diagnosis often involves multiple faults and different vehicle models. At the End-of-Line test in an automobile assembly plant, engineers need to diagnose over 100 different types of faults including vacuum leak, idle quality, axle ratio, transmission shift, etc. on hundreds of different vehicle models. The fuzzy model we developed attempts to automate this engineering diagnosis by using fuzzy logic. The fuzzy model is designed so that it can be implemented to detect any one particular type of fault. Figure 1 presents the overview of the fuzzy diagnostic model. Figure 1(a) shows the system components, an inference engine, a fuzzy rule generator and a membership function (MBF) optimizer. Figure 1(b) illustrates the fuzzy knowledge base which is composed of the fuzzy rules and the fuzzy membership functions (MBFs). The fuzzy diagnostic knowledge can be generated either from engineer experts or training data through a machine learning algorithm. The system kernel interacts with the knowledge base directly during the fuzzy inference. Figure 1(c)

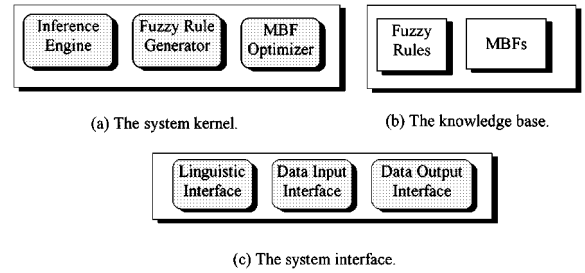

Figure 1. A fuzzy model for engineering diagnosis.

illustrates the system interface. The system interface provides a linguistic interface between the engineer experts and the knowledge base, and a data input/output interface between the user and the system kernel. The focus of this paper is to present our research results in fuzzy rule generation and membership function optimization.

Fuzzy reasoning is performed within the context of a fuzzy system model, which consists of control, solution variables, fuzzy sets, proposition (rule) statements, and the underlying control mechanisms that tie all these together into a cohesive reasoning environment. The fuzzy rules can be completely characterized by a set of control variables, $X = \{x_1, x_2, \ldots, x_n\}$ and a solution variable. Each control variable is associated with a set of fuzzy terms $\Sigma_i = \{\alpha_i^1, \ldots, \alpha_i^{p_i}\}$.

In this fuzzy fault diagnosis model, the control variables are the parameters that reflect the faulty behavior the system is responsible to detect. The fuzzy model has one solution variable y, which describes the particular type of the fault the fuzzy system is responsible to detect. Solution variable y is associated with fuzzy terms $\Gamma = \{\tau_1, \ldots, \tau_q\}$.

Each fuzzy rule has the format:

$$\begin{aligned}&\textbf{IF } \left(x_{k1} \text{ is } \alpha_i^{k1}\right) \text{ AND } \left(x_{k2} \text{ is } \alpha_i^{k2}\right)\\ &\quad \text{AND } \ldots \left(x_{km} \text{ is } \alpha_i^{km}\right),\\ &\textbf{THEN } y \text{ is } \tau_i^k\end{aligned}$$

where $m \leq n, x_{k1}, x_{k2}, \ldots, x_{km} \subset X$, $\{\alpha_i^{k1}, \alpha_i^{k2}, \ldots, \alpha_i^{km}\} \subset \Sigma_i$, and $\tau_i^k \in \Gamma$. Within this application domain, unconditional rules are not necessary.

In fuzzy logic, each control and solution variable is associated with a set of fuzzy membership functions each of which corresponds to a fuzzy term. A membership function of a control variable is a control surface that responds correctly to a set of expected data points. In general, the membership functions associated with a fuzzy variable can be defined by a set of **critical parameters** that uniquely describe the characteristics of the membership functions. For example, if the membership functions are Gaussian, the standard deviation and the mean of each Gaussian function constitute the critical parameters; if the membership functions are triangular functions, the locations of the triangular apices along with the starting and the ending points form the critical parameters (see Fig. 2). The characteristic of an inference engine is largely affected by these critical parameters. Given the test data and the fuzzy rules, assigning different values to the critical parameters usually makes the inference engine output different classification results. Similarly, given the training data, assigning different values to the critical parameters usually makes the rule generator generate different fuzzy rules.

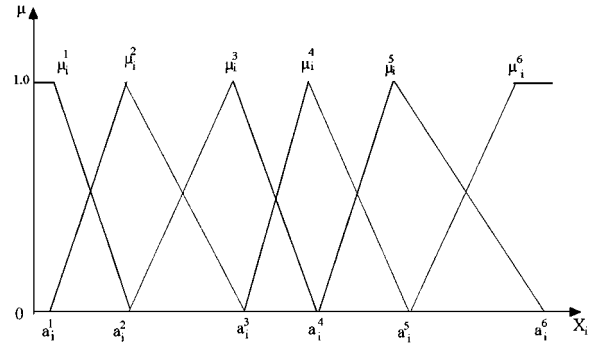

Figure 2. Critical parameters of the fuzzy membership functions of a control variable.

Therefore, fuzzy rule generation and membership functions are critical to the performance of a fuzzy system.

3. Fuzzy Rule Generation

A complete set of fuzzy rules contains n^m fuzzy rules, where n is the number of fuzzy terms and m is the number of fuzzy parameters. A fuzzy rule generation can be computationally expensive [8–10, 13, 14]. A number of fuzzy rule generation algorithms have been published. Katayama et al. [21] described a gradient-based method to tune fuzzy membership functions and to generate fuzzy rules. The gradient descent optimization algorithm requires the objective function to have continuous first-order derivatives. Unfortunately, the input-output function of a fuzzy system usually does not satisfy this condition for two reasons. First the `min-max` operations used in fuzzy implication are not continuous, and secondly the triangular functions do not have continuous derivatives. Others attempted to use genetic algorithm, neural network and fuzzy-C mean to generate fuzzy knowledge [14, 17, 20, 22, 23]. The algorithm we present here efficiently generates a compact and optimal set of fuzzy rules. In this algorithm, fuzzy rule generation is combined the rule with fuzzy membership optimization.

In fuzzy logic, each variable is associated with a set of fuzzy membership functions each of which corresponds to a fuzzy term. A membership function of

a control variable is a control surface that responds to a set of expected data points. As we discussed in Section 2 that the membership functions associated with a fuzzy variable can be collectively defined by a set of **critical parameters** that uniquely describe the characteristics of the membership functions and the characteristic of an inference engine is largely affected by these critical parameters. The possible values of these critical parameters form a hyper space and the system response to the control parameters form a control surface. The optimization of fuzzy membership functions is to find a point in the hyper space of the critical parameters that makes the control surface (the system response) to react correctly to a set of training data.

At the beginning of the algorithm, the critical parameters of the fuzzy membership functions are initialized at an arbitrary point, and are further optimized during and after the generation of the fuzzy rules. The fuzzy rules and the membership functions are generated and optimized in an iterative fashion.

Fuzzy rules are generated based on the concept of "**the dominant rule of a data sample**". We use an example to explain this concept. Assume we have rule r "*if* x_1 *is* L *and* x_2 *is* M, *then* y *is* H" where, x_1 and x_2 are control variables, y is a solution variable, and L, M and H are fuzzy terms. For a data sample s, if its belief value for control variable x_1 in fuzzy term L is larger than the belief value for x_1 in any other fuzzy terms, and its belief for value control variable x_2 in fuzzy term M is larger than its belief value for x_2 in any other fuzzy terms, then fuzzy rule r is called **the dominant rule** of the data sample s and the sample s is called **a dominated sample of rule** r. Although input sample s may fire other rules in the knowledge base, its belief values for other fuzzy rules must be lower than its belief value for rule r since its belief values for a and b in fuzzy term L and M, respectively, are higher than any other fuzzy terms. Therefore, the fuzzy inference result of s is dominated by the rule r.

Using this method, the algorithm generates a dominant rule for each data sample in the training set. If the newly generated dominant rule does not exist, it is added into the rule base. If the rule already exists, the algorithm increases the priority of the rule by adding the priority value of the new rule to the existing one. If the new rule is conflicting with a previously generated rule, the algorithm decreases the priority of the existing rule by subtracting the priority of the new rule from this existing one. If the priority of an existing rule becomes negative, the rule is replaced with the new one. Note the priority value of a fuzzy rule is normalized using the number of data samples in the training set. Therefore, the priority of a rule is always less than 1.

We use an example to illustrate this step of rule generation. Table 1 shows all the rules generated by 1000 samples in a training set. We have two control variables, x_1 and x_2 and a solution variable y. The data samples in the training set form clusters in such a way that data samples belonging to one cluster have the largest belief values in the same fuzzy terms for x_1 and x_2, and for the solution variable y. For example, cluster x_1 is *LOW* and x_2 is *LOW* has 200 support

Table 1. The data in the table shows the rule generation in one iteration. The fuzzy rules that will be generated are shown in bold.

	Dominant rule							
			Consequence					
	Antecedent		y is low		y is medium		y is high	
Cluster	x_1	x_2	No. of sample	Average belief value	No. of sample	Average belief value	No. of sample	Average belief value
0	Low	Low	60	0.72	**100**	**0.80**	40	0.65
1	Medium	Low	20	0.62	**150**	**0.77**	30	0.67
2	High	Low	30	0.75	**140**	**0.81**	30	0.81
3	Low	Medium	**120**	**0.74**	20	0.72	20	0.63
4	Medium	Medium	**100**	**0.79**	20	0.69	10	0.77
5	High	Medium	0	—	0	—	0	—
6	Low	High	0	—	0	—	0	—
7	Medium	High	0	—	0	—	0	—
8	High	High	10	0.65	10	0.73	**90**	**0.71**

Table 2. Fuzzy rules extracted from Table 1.

	Dominant rule			
	Antecedent		Consequence	
Cluster	x_1	x_2	y	Priority
0	Low	Low	Medium	0.04540
1	Medium	Low	Medium	0.09925
2	High	Low	Medium	0.09000
3	Low	Medium	Low	0.07530
4	Medium	Medium	Low	0.06825
5	High	Medium	—	—
6	Low	High	—	—
7	Medium	High	—	—
8	High	High	High	0.05700

samples, which are called **the dominated samples** of this will-be-generated rule, and this rule is called **the dominant rule**. The consequence of this dominant rule has three possibilities, y is *Low* which is supported by 60 data samples; y is *Medium* which is supported by 100 data samples; and y is *High*, which is supported by 40 data samples. Note each cluster is associated with an attribute called average belief value. For example, the average belief value for each of the three clusters we just discussed is 0.72, 0.80 and 0.65, respectively. Using Winner-Take-All (WTA) based on the average belief value, we can generate a fuzzy rule, x_1 is *LOW* and x_2 *is LOW* and y *is Medium*. According to the priority rule generation, we get fuzzy rules and their priorities in Table 2 from the clusters in Table 1. The priority of each rule is computed in such a way that its value shows how accurately it describes the training data. Mathematically, the priority of a fuzzy rule is computed as follows:

$$\frac{\#\_samples_w * A\_B\_V_w - \frac{1}{k}\sum_{j=1}^{k} \#\_samples_l^j * A\_B\_V_l^j}{n}$$

where $A\_B\_V_w$ is the average belief value of the winning rule, $A\_B\_V_l^j$ is the average belief value of the jth losing rule, and k is the number of losing rules, and n is the number of total samples in the training set.

For example, the priority of the rule extracted from first row in Table 1 is computed as follows:

$$(100 \times 0.80 - (60 \times 0.72 + 40 \times 0.65) \div 2) \div 1000 = 0.0454.$$

After the fuzzy rules are generated, the belief value of a data sample firing a fuzzy rule will be used in reclustering as the weight (or probability) of belonging to this cluster. So the membership functions are recomputed then used to generate new fuzzy rules.

For every newly extracted rule, the algorithm attempts to merge it with the rules already in the rule table. If a merge takes place, the redundant control variables are pruned from the antecedent of the newly-extracted rule, and the resulting "no. of support samples" for the new rule is the sum of the "no. of support samples" of all the merged rules. In Table 2, the consequences of rule #5, #6 and #7 are null because no sample belongs to these clusters. These **null rules** are very useful in the rule reduction process. For example in Table 2, without rule #5 $\sim$ #7, only rule #0 $\sim$ #2 in Table 2 can be merged:

Rule #0 $\sim$ #2 $\rightarrow$ "if $\mathbf{x_2}$ is `LOW`, then $\mathbf{y}$ is `MEDIUM`"

"No. of support samples" of the new rule is $100 + 150 + 140 = 390$. With the null rules (#5 $\sim$ #7), rule #3 $\sim$ #8 in Table 2 can also be merged:

Rule #3 $\sim$ #5 $\rightarrow$ "if $\mathbf{x_2}$ is `MEDIUM`, then $\mathbf{y}$ is `LOW`"

Rule #6 $\sim$ #8 $\rightarrow$ "if $\mathbf{x_2}$ is `HIGH`, then $\mathbf{y}$ is `HIGH`"

"No. of support samples" of the two new rules are: $120 + 100 + 0 = 220$ and $0 + 0 + 90 = 90$, respectively. The final fuzzy rules extracted in this iteration are listed in Table 3, note all the null rules are removed.

Unlike a fuzzy-neural network in which the fuzzy rules are encoded in a distributed manner among the interconnection weights, the internal representation of the fuzzy rules in our system is a look-up table. The look-up table representation provides the flexibility, meanwhile it has the least memory requirement. With this representation, the system speed benefits from the situation of less fuzzy rules present because less items in the rule table need to be searched, while the speed

Table 3. The final fuzzy rules extracted in current iteration.

	Merged dominant rule			
	Antecedent		Consequence	
Rule	x_1	x_2	y	Priority
0	*Don't care*	Low	Medium	0.23465
1	*Not* High	Medium	Low	0.14355
2	High	High	High	0.05700

of a fuzzy-neural network system does not decrease unless the neural network has been reconstructed. We can further improve the system speed by sorting the rule table according to the "no. of support samples" of the rules because the table is searched sequentially and the rules fired most frequently are placed at the head of the table.

The next step in the rule generation algorithm is reclustering and calculation of the location of each cluster centers in the algorithm. The reclustering step attempts to solve the problem such that the training samples belonging to different classes may be included in the same cluster in the previous clustering step. Thus, the rule extracted from such a cluster will not be effective. This is similar to the problem encountered in the fuzzy c-means algorithm. In the fuzzy c-means algorithm, the data points always belong to the nearest clusters, and the data points belonging to different classes are not discriminated while calculating the locations of the cluster centers.

The reclustering procedure uses a measure of distance between a sample and a cluster. Without losing generality, we assume a fuzzy rule is in the following form:

"IF (x_1 is c_1 and x_2 is c_2 and ... and x_n is c_n),
THEN (y is c)."

For class c_i of the control variable x_i, there exists a range (x_{icL}, x_{icH}) in which the membership function of class c_i has the largest value. Therefore, class c of the solution variable is mapped onto a n-dimensional **hyper-cubed** in the input space by this rule. The n-dimensional hyper-cubed is specified by a set of ranges (x_{icL}, x_{icH}), where $i = 1, 2, \ldots, n$. All the samples inside a hyper-cubed is called a **cluster**. The squared distance between sth sample and a cluster is defined as follows:

$$SD = \sum_{i=1}^{n} \varepsilon_{xi}(s),$$

where

$$\varepsilon_{xi}(s) = \begin{cases} \left(x_{icL} - x_i^{(s)}\right)^2 & \text{if } \left(x_i^{(s)} < x_{icL}\right) \\ 0 & \text{if } \left(x_{cL} < x_i^{(s)} < x_{icH}\right) \\ \left(x_i^{(s)} - x_{icH}\right)^2 & \text{if } \left(x_{icH} < x_i^{(s)}\right) \end{cases}$$

After all the training samples have been reclustered, the location of every cluster center is calculated. If the locations of these clusters are close to the critical parameters of the MBFs, the fuzzy rule generation algorithm ends. Otherwise, the critical parameters of fuzzy membership functions are updated to the locations of the cluster centers, the iteration of fuzzy rule extraction, reclustering, and MBFs updating repeats.

In the implementation of the fuzzy rule generation algorithm, we used a heap structure to store all the rules. The rules with the lower priority are placed on the top of the structure. This structure has two advantages. First, the fuzzy rule with the least priority can be discarded easily, and secondly, since all fuzzy rules are sorted according to their priority in this heap structure, the search based on priority values can be implemented more efficiently.

4. Fuzzy Inference Using Priority Based Fuzzy Rules

The fuzzy rules generated by the algorithm described in Section 3 is a compact subset of the complete fuzzy rule set, which is determined by the control and solution variables and their associated fuzzy terms. Since the fuzzy rule pruning process eliminates unreliable fuzzy rules, and the rule merging process combines several rules into one, the resulting compact fuzzy rule knowledge base allow the system to perform robust and efficient detection. However, during the fuzzy inference, it is possible that an input data sample fires no rule in the knowledge base. In order to deal with this problem, we developed the following inference scheme that fires the nearest rule to the input sample.

Generally, a fuzzy rule can be considered as a fuzzy cluster in the input space. For instance, a fuzzy rule written as

if x_1 *is* ***LOW*** *and* x_2 is ***HIGH***, *then y is* ***MEDIUM***

represents a fuzzy cluster centered at the ***center point*** of the fuzzy membership functions of "$\boldsymbol{x}_1$ is **LOW**" and "$\boldsymbol{x}_2$ is **HIGH**". Based on this concept, we define the following distance measure between a data sample and a fuzzy rule. Let an input data sample be $I = \{a_1, \ldots, a_n\}$, where a_i is the instantaneous value of fuzzy variable $x_i, i = 1, \ldots, n$, $\Sigma = \{\alpha_1, \alpha_2, \ldots, \alpha_p, \phi\}$ is a set of fuzzy terms associated with each control variable in X, ϕ is a symbol that serves as "*don't care*". With the introduction of ϕ, a fuzzy rule can always be written in the following

general form:

if x_1 is α_{1k} and x_2 is α_{2k} and ... and x_n is α_{nk}
then z is β

where $\alpha_{ik} \in \Sigma$, for $i = 1, \ldots, n$, z is a solution variable and β is a fuzzy term. For example, for a system has three control variables, $\{x_1, x_2, x_3\}$, and the fuzzy terms are {LOW, MEDIUM, HIGH} if we have a fuzzy rule

if x_1 *is* ***LOW*** *and* x_3 *is* ***HIGH***, *then* y *is* ***MEDIUM***,

we can rewrite the rule equivalently as

if x_1 *is* ***LOW*** *and* x_2 *is* ϕ *and* x_3 *is* ***HIGH***,
then y *is* ***MEDIUM***

The *distance* between a data sample I and a fuzzy rule k is defined as

$$d = \sqrt{\sum_{i=1}^{n} (a_i - c_{ik})^2}$$

where c_{ik} is the center point of the fuzzy membership function of fuzzy variable x_i for fuzzy term $\alpha_{ik} \neq \phi$, otherwise $c_{ik} = a_i$, which sets $(a_i - c_{ik}) = 0$. The fuzzy rule that has the shortest distance to I is fired.

The priority based fuzzy rule system also provides a convenient way to incorporate multiple knowledge sources at the inference stage. For example, an intelligent system may have multiple sources of knowledge, engineering expert knowledge, knowledge generated from training data sets, knowledge obtained from scientific laws, etc. The fuzzy inference engine can assign different priorities to these knowledge sources, and the summation of the priorities over the knowledge sources should be 1. During the inference, this knowledge source priority is multiplied to the rule priority as the weight of a fired rule. This concept is outlined as follows.

This rule priority technique also enables a system to learn knowledge in an accumulative fashion. The idea behind the accumulative learning is that we may begin with a set of rules that were generated based on either expert knowledge or an insufficient data set, and then we may encounter data samples that represent new knowledge during the fuzzy inference stage. If the inference engine notice that a large number of data samples match no existing rules, it may want to generate new rules, which are accumulative to the original rules. We developed an accumulative learn algorithm that generates fuzzy rules from a new set of data. The algorithm assumes the fuzzy rules in the initial set have priority values. First, the accumulative algorithm assigns a new priority value to each fuzzy rule by formula: $p * \frac{N}{N+M}$, where p is the old priority value of the rule, and M is the number of data samples in the current training set. The algorithm then extracts new rules from the new training data using the fuzzy rule generation algorithm described above. The priority of a new rule is computed by $\frac{1}{N+M} \sum_{s=1}^{M} b_s$, where b_s is the belief value that sth data sample in the new training set that fires this rule. If a new rule does not exist, it is added into the knowledge base. If a new rule is identical to an existing rule, we add the priority of the new rule to the priority of the existing rule. If a new rule is conflicting with an existing rule, we compare the priority values of the two rules. If the priority of the new rule is less than the priority of the old rule, we decrease the priority of the old rule by subtracting the priority of the new rule from the priority value of the old rule. Otherwise, the old rule will be replaced by the new rule whose new priority is the difference between the its priority and the priority of the existing rule.

The concept of accumulative learning is very useful in many applications in which knowledge about an event is not available during the fuzzy learning stage. For example, the End-of-Line test in automotive assembly plants is required to test new vehicle models frequently, and there is usually no data available for learning at the beginning of the production. However, we can have an initial fuzzy knowledge base generated by engineering expert or by the data samples of similar vehicle models. When more data become available, the accumulative learning algorithm can generate a more reliable knowledge base. Note in the accumulative learning, we can emphasize the reliability of the initial knowledge base by adjusting the priority values of its fuzzy rules.

5. A Fuzzy Automotive Engineering Diagnostic System

The fuzzy rule generation algorithm described in the previous sections has been implemented in an automotive engineering diagnosis system for the End-of-Line test in automobile assembly plants. Figure 3 shows a distributed diagnostic system for the End-of-Line test.

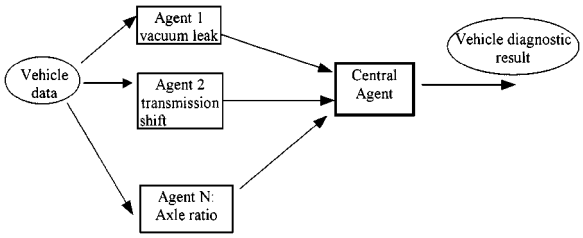

Figure 3. A distributed diagnostic system.

The distributed diagnostic system employs a number of diagnostic agents each of which is responsible for diagnosing one particular type of faulty behavior. The central agent identifies the minimum set of components that give rise to the faulty behaviors of the testing vehicle reported by the diagnostic agents. In general speaking, the diagnostic agents have the local knowledge, the knowledge that is sufficient for each agent to perform its own diagnostic task. The central agent has the global knowledge, the knowledge of the entire machine under diagnosis and the knowledge of the relationship between the faulty behaviors of the diagnostic agents.

Every diagnostic agent is implemented using the fuzzy model illustrated in Fig. 1. The fuzzy rules that are used by each diagnostic agent are generated using the algorithm described in the previous section. We will use the agent for vacuum leak detection as an example to illustrate our fuzzy rule generation algorithm.

To model the vacuum leak in the Electronic Engine Control, we first need to determine what the control variables are. One Electronic Engine Controller parameter that can give insight into the vacuum leak problem is the desired air/fuel flow 'Lambda' (λ). The air/fuel flow is measured by one or two oxygen sensors (HO2S) depending on whether it is a mono-HO2S or stereo HO2S system. A stereo system controls fuel separately from cylinder bank to cylinder bank e.g., LAMBSE1 is for cylinder bank #1 and LAMBSE2 is for cylinder bank #2. If there is a vacuum leak, more air is entering the combustion process than is calculated to be flowing past the air meter, causing the combustion to burn lean. In this case the value of LAMBSE will be rich ($\lambda < 1$). If the air flow sensor is reading more air than is actually entering the combustion process, less air is entering the combustion process than is calculated from the air flow sensor; causing the combustion to burn rich. In this case, the value of LAMBSE will be lean ($\lambda > 1$). Similarly, any engine component that affects air, fuel, or spark will have an impact on the calculated value of LAMBSE. However, LAMBSE failures can be generated by, in addition to engine vacuum leaks, a number of other factors including HO2S failures, fuel systems problems, intake manifold problems, ignition problems, etc. In a stereo HO2S vehicle, we receive two channel data, i.e., HO2S-bank 1 and HO2S-bank 2. In this type of system, LAMBSE failures at one side or both sides may indicate different problems. The detection of vacuum leak in an engine is a complicated problem with a number of uncertainty factors, and knowledge on this problem is not complete. Therefore, the fuzzy system is a good solution for detecting engine vacuum leak. In addition to Lambsel and Lambse2, three other parameters are useful in detecting vacuum leak, throttle position, idle speed DC, and mass air flow. Te amount of air entering a gasoline engine is controlled by a throttle plate that opens and closes. A throttle position sensor senses the open angle of the throttle and transforms it into a voltage that is read by the vehicle controller. If there is a vacuum leak, the throttle position should be low. The idle speed duty cycle refers to the pulse train that modulates the solenoid. The mass air flow tells us the mass of air entering the engine. Therefore, the fuzzy vacuum detection agent has five control variables, {throttle position, Lambse_1, Lambse_2, Idle speed DC, Mass air flow}, and one solution variable, {vacuum_leak}. Each fuzzy variable is associated with three fuzzy terms {LOW, MEDIUM, HIGH}. Figure 4 shows the block diagram of the vacuum detection agent system.

The fuzzy knowledge base contains fuzzy rules and membership functions obtained using the methods described in the previous section. We have trained and tested the fuzzy vacuum diagnostic system on two different vehicle models. In order to protect proprietary information, the names of these vehicle models are not reported, we simply call them vehicle model I and vehicle model II. We downloaded vehicle log files directly from test sites of the Ford Motor Company. Vehicle samples in the log files were examined by test

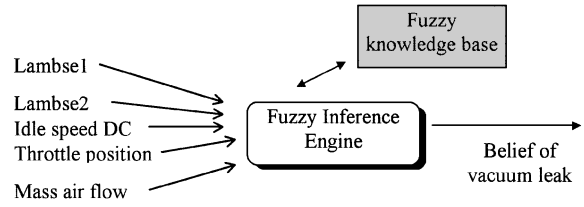

Figure 4. A fuzzy agent for vacuum leak diagnosis.

Table 4. Data sets used in system learning and testing.

	Vehicle model I	Vehicle model II
Training		
Good car samples	8298	7736
Bad car samples	1536	2560
Test		
Good car sample	12447	11605
Bad car sample	9	15

engineers and marked with comments such as "Failed", "Passed", "air leak," "vacuum leak," etc. We extracted vehicle samples from these files and separated them based on the comments in the vehicle log files into good samples and bad samples that mean the vehicles have a vacuum leak problem. For each of the two vehicle models, we have one training set and one test set. The detail of the data for both vehicle models is listed in Table 4. A good car sample means "vacuum leak is LOW," a bad car sample means "vacuum leak is HIGH." The experiment results are reported as follows.

For vehicle model I, the engineer-experts at an automotive company provided six fuzzy rules and three membership functions for each control variable, which are listed in Tables 5 and 6, respectively. The inference accuracy using this knowledge base is 50% for bad vehicles and 29.4% for the training set, and 17.5% and 27% on the good and bad samples, respectively, in the test set.

The fuzzy rules and the membership functions generated by our machine learning algorithms from the training data are shown in Tables 6 and 7, respectively. The test result in the training set on good vehicles is 99.4% in accuracy, and 100% in accuracy for bad vehicles. For the test set, the system detected 99.5% good vehicles and 100% on the test vehicles.

For vehicle model II, we conducted the following experiments. First, we used the fuzzy knowledge base generated from vehicle model I to test on the data obtained from vehicle model II. When the fuzzy rules in Table 7 and the membership functions in Table 8 are used to process the vehicle model II data files, the results were very poor. The detection accuracy is 22.7% for bad vehicles and 22.9% for good vehicles. This confirms our assumption: different vehicle models are engineered differently; therefore, for each vehicle model, we need to train the system on the data acquired from the vehicles of the same model.

The fuzzy rules and the membership functions generated by our machine learning algorithms on the training data of model II are shown in Tables 9 and 10, respectively. The system performance using this set of fuzzy knowledge is excellent. On the training set, the system detected correctly good and bad vehicles 100%,

Table 5. The fuzzy rules summarized from human knowledge for vehicle model I.

Rule no.	Antecedent					Consequence
	Lambse1	Lambse2	Idle speed	Mass air flow	Throttle position	Vacuum leak
0	LOW	LOW	LOW	MEDIUM	LOW	HIGH
1	LOW	LOW	LOW	MEDIUM	MEDIUM	HIGH
2	LOW	MEDIUM	*anything*	*anything*	*anything*	LOW
3	LOW	HIGH	*anything*	*anything*	*anything*	LOW
4	MEDIUM	LOW	*anything*	*anything*	*anything*	LOW
5	HIGH	LOW	*anything*	*anything*	*anything*	LOW

Table 6. Membership functions summarized from human knowledge for vehicle model I.

Critical parameter no.	Control variables					Solution variable
	Lambse1	Lambse2	Idle speed	Mass air flow	Throttle position	Vacuum leak
0	0.8	0.8	0.2	0	150	0
1	0.95	0.95	0.34	0.75	200	0.5
2	1.1	1.1	0.44	2	250	1

Table 7. The fuzzy rules generated through machine learning for vehicle model I.

Rule no.	Antecedent					Consequence
	Lambse1	Lambse2	Idle speed	Mass air flow	Throttle position	Vacuum leak
0	HIGH	HIGH	MEDIUM	HIGH	MEDIUM	LOW
1	HIGH	HIGH	MEDIUM	HIGH	HIGH	LOW
2	HIGH	HIGH	HIGH	HIGH	MEDIUM	LOW
3	MEDIUM	MEDIUM	HIGH	HIGH	HIGH	HIGH
4	LOW	LOW	LOW	HIGH	HIGH	HIGH
5	LOW	LOW	LOW	MEDIUM	HIGH	HIGH
6	LOW	LOW	LOW	MEDIUM	MEDIUM	HIGH
7	LOW	LOW	HIGH	MEDIUM	HIGH	HIGH
8	HIGH	HIGH	HIGH	HIGH	HIGH	LOW
9	MEDIUM	HIGH	HIGH	HIGH	HIGH	HIGH
10	MEDIUM	MEDIUM	MEDIUM	HIGH	HIGH	LOW
11	HIGH	HIGH	HIGH	MEDIUM	HIGH	LOW
12	MEDIUM	MEDIUM	MEDIUM	MEDIUM	HIGH	HIGH
13	LOW	LOW	MEDIUM	MEDIUM	HIGH	HIGH
14	LOW	MEDIUM	LOW	MEDIUM	HIGH	HIGH
15	MEDIUM	MEDIUM	HIGH	LOW	HIGH	HIGH
16	HIGH	LOW	HIGH	HIGH	MEDIUM	HIGH
17	MEDIUM	MEDIUM	MEDIUM	HIGH	MEDIUM	HIGH
18	*anything*	LOW	HIGH	HIGH	LOW	HIGH

Table 8. Membership functions generated through machine learning for vehicle model I.

Critical parameter no.	Control variables					Solution variable
	Lambse1	Lambse2	Idle speed	Mass air flow	Throttle position	Vacuum leak
0	0.748385	0.737143	0.213	0.009	156	0
1	0.828	0.808375	0.337143	0.7645	191.783	0.5
2	0.995903	0.97529	0.3938	0.89245	201.828	1

on the test set, the system correctly detected good vehicles 99.99% and bad vehicles 100%. We have studied the same problem using three neural network architectures, multilayered Backpropagation (BP) network, Radial Basis Function (RBF), and Fuzzy Adaptive Resonance Theory (ART) network [23]. The fuzzy system described above out performed both the BP and RBF network. It has comparable with Fuzzy ART network.

6. Conclusions

In this paper, we presented a fuzzy diagnostic model for automotive fault diagnosis and the fuzzy rule generation algorithm used in the fuzzy model. The fuzzy diagnostic model is based on priority rules. The model provides an efficient and robust fuzzy rule generation algorithm. It uses priority based fuzzy inference that is capable of integrating multiple sources of knowledge, accumulative learning, and firing the best rules in the knowledge base. The fuzzy diagnostic model has been implemented in a vacuum leak detection agent system. The vacuum leak detection agent system has been tested on large sets of vehicle data samples downloaded directly from automotive assembly plants and the performance is proven to be excellent. Currently, the fuzzy diagnostic system in the process of being integrated into a Ford test system for end-of-line test.

Table 9. Fuzzy rules generated through machine learning for vehicle model II.

Rule no.	Antecedent					Consequence
	Lambse1	Lambse2	Idle speed	Mass air flow	Throttle position	Vacuum leak
0	HIGH	MEDIUM	LOW	*anything*	LOW	LOW
1	HIGH	MEDIUM	LOW	MEDIUM	MEDIUM	LOW
2	HIGH	MEDIUM	LOW	MEDIUM	HIGH	LOW
3	HIGH	MEDIUM	LOW	LOW	HIGH	LOW
4	HIGH	MEDIUM	LOW	LOW	MEDIUM	LOW
5	*anything*	LOW	LOW	LOW	LOW	HIGH
6	LOW	MEDIUM	MEDIUM	MEDIUM	HIGH	HIGH
7	MEDIUM	LOW	LOW	LOW	MEDIUM	HIGH
8	LOW	MEDIUM	MEDIUM	HIGH	HIGH	LOW
9	LOW	LOW	MEDIUM	MEDIUM	HIGH	HIGH
10	MEDIUM	LOW	LOW	MEDIUM	HIGH	HIGH
11	HIGH	LOW	MEDIUM	HIGH	MEDIUM	HIGH
12	LOW	HIGH	MEDIUM	MEDIUM	MEDIUM	HIGH
13	HIGH	MEDIUM	MEDIUM	MEDIUM	MEDIUM	LOW
14	MEDIUM	LOW	MEDIUM	MEDIUM	MEDIUM	HIGH
15	LOW	LOW	MEDIUM	MEDIUM	MEDIUM	HIGH
16	LOW	MEDIUM	LOW	MEDIUM	MEDIUM	HIGH
17	LOW	LOW	LOW	MEDIUM	MEDIUM	HIGH
18	LOW	LOW	HIGH	HIGH	LOW	HIGH
19	LOW	MEDIUM	MEDIUM	HIGH	LOW	LOW
20	MEDIUM	LOW	MEDIUM	HIGH	LOW	HIGH
21	LOW	LOW	MEDIUM	HIGH	LOW	HIGH
22	MEDIUM	LOW	LOW	HIGH	LOW	HIGH
23	LOW	LOW	MEDIUM	MEDIUM	LOW	HIGH

Table 10. Membership functions generation through machine learning for vehicle model II.

Critical parameter no.	Control variables					Solution variable
	Lambse1	Lambse2	Idle speed	Mass air flow	Throttle position	Vacuum leak
0	0.754917	0.816889	0.31	0.718059	188.731	0
1	0.862143	0.968392	0.383846	0.795933	197.63	0.5
2	0.963383	1.211	0.55	0.951143	205.588	1

Acknowledgment

This work is supported in part by a Grant from NSF DMII-9612190 and a grant from the Ford Motor Company.

References

1. Harry Tennant, "Onboard knowledge systems in vehicles," *Vehicle Electronics in the 90's: Proceedings of the International Congress on Transportation Electronics*, October 1990.
2. Filljov, M. Marinov, and S. Ovcharov, "Engine diagnostic expert system," in *The Eighteenth International Symposium on Automotive Technology and Automation*, May–June 1988.
3. Johan de Kleer, "Focusing on probable diagnoses," *AAAI-91*, pp. 842–848, 1991.
4. Johan de Kleer and Brian C. Williams, "Diagnosing multiple faults," *Artificial Intelligence*, vol. 32, pp. 97–130, 1987.
5. L.S. Tedesco, "Service Bay Dragnostic System," Ford Motor Co., Society of Automotive Engineers, paper number 861030, 1986.

6. Brennan T. Hamilton and Yi Lu, "Diagnosis of automobile failures using fuzzy logic," *The Eighth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Melbourne, Australia, June 1995.
7. Zheng Xiaojun, Yang Shuzi, Zhou Anfa, and Shi Hanmin, "A knowledge-based diagnosis system for automobile engines," *The International Journal of Advanced Manufacturing Technology*, 1988.
8. M. Ayoubi, "Neuro-fuzzy structure for rule generation and application in the fault diagnosis of technical processes," in *Proc. American Control Conference*, Seattle, USA, 1995, pp. 2757–2761.
9. H. Nomura, H. Ichihashi, and T. Watanabe, "A self-tuning method of fuzzy reasoning control by descent method," in *Proc. of 4th IFSA Congress*, Brussels, 1991, vol. Eng., pp. 155–158.
10. F.C.-H. Rhee and R. Krishnapuram, "Fuzzy rule generation methods for high-level computer vision," *Fuzzy Sets and Systems*, vol. 60, pp. 245–258, 1993.
11. T. Takagi and M. Sugeno, "Fuzzy idnetification of systems and its applicaitons to modeling and control," *IEEE Trans. Syst., Man, and Cybern.*, vol. SMC-15, no. 1, pp. 116–132, 1985.
12. Alanna Quail and Adnan Shaout, "State-of-the-art in household applicances using fuzzy logic," in *Second International Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems*, College Station, TX, December 1992.
13. R. Katayama, Y. Kajitani, and Y. Nishida, "A self-generating and tuning method for fuzzy modeling using interior penalty method," in *Proc. Second Int. Conf. Fuzzy Logic and Neural Networks*, Iizuka, Japan, 1992, pp. 17–22.
14. H. Kang, "An automated rule design of fuzzy logic controllers for uncertain dynamic systems," in *Second IEEE Int. Conf. on Fuzzy Systems*, 1993, pp. 261–266.
15. C.K.P. Chu and J.M. Mendel, "First break refraction event picking using fuzzy logic systems," *IEEE Transactions on Fuzzy Systems*, vol. 2, pp. 255–266, November 1994.
16. P. Eklund, J. Forsström, A. Holm, M. Nyström, and G. Selén, "Rule generation as an alternative to knowledge acquisition: A systems architecture for medical informatics," *Fuzzy Sets and Systems*, vol. 66, pp. 195–205, 1994.
17. S. Mitra and S.K. Pal, "Fuzzy multi-layer perceptron, inference and rule generation," *IEEE Trans. Neural Networks*, vol. 6, pp. 51–63, 1995.
18. E. Tazaki and N. Inoue, "A generation method for fuzzy rules using neural networks with planar lattice architecture," in *Proc. IEEE Int. Conf. Neural Networks*, Piscataway, USA, 1994, pp. 1743–1748.
19. I.B. Turksen and H. Zhao, "An equivalence between inductive learning and pseudo-Boolean logic simplification: A rule generation and reduction scheme," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 907–917, 1993.
20. T.W. Cheng, D.B. Goldgof, and L.O. Hall, "Fast clustering with application to fuzzy rule generation," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Yokohama, Japan, 1995, pp. 2289–2295.
21. Ryu Katayama, Yuji Kajitani, and Yukiteru Nishida, "A self generating and tuning method for fuzzy modeling using interior penalty method and its application to knowledge acquisition of fuzzy controller," *Fuzzy Control Systems*, pp. 198–224, 1993.
22. H. Ishigami, Y. Hasegawa, T. Fukuda, and T. Shibata, "Automatic generation of hierarchical structure of fuzzy inference by genetic algorithm," in *IEEE Int. Conf. on Fuzzy Systems*, 1994, pp. 1566–1570.
23. J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plemun Press: New York, 1981.
24. Yi Lu, Hong Guo, and Lee Feldkamp, "Robust neural learning from unbalanced data samples," *IEEE IJCNN*, 1998.
25. S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 721–741, 1984.
26. H. Genther, A. König, and M. Glesner, "Rule weight generation for a fuzzy classification system based on fuzzy clustering methods," in *Proc. 3rd IEEE Int. Conf. Fuzzy Syst.*, Orlando, USA, 1994, pp. 614–617.
27. L.O. Hall, A.M. Bensaid, L.P. Clarke, R.P. Velthuizen, M.S. Silbiger, and J.C. Bezdek, "A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain," *IEEE Trans. Neural Networks*, vol. 3, pp. 672–682, 1992.

Yi Lu received a M.S. degree in computer science from Wayne State University, Detroit, Michigan, in 1983 and Ph.D. degree in Computer, Information and Control Engineering from the University of Michigan, Ann Arbor, Michigan, in 1989.

From 1989 to 1992, she was a research scientist at the Environmental Research Institute of Michigan, Ann Arbor, Michigan. Currently she is an associate professor at the University of Michigan-Dearborn. Her research interests include computer vision, neural networks and fuzzy logic.

Dr. Lu is a senior member of IEEE Computer Society and a member in the American Association of Artificial Intelligence. She is an associate editor for the Journal of Pattern Recognition.

Tie-Qi Chen got his Bachelor's degree in theoretical physics at Department for Intensive Undergraduate Instruction, Nanjing University, Nanjing, China in 1989. He received his Ph.D. in optical information processing at Physics Department, Fudan University, Shanghai, China in 1994. Since 1995, he has been working as a visiting scholar at Department of Electrical and Computer Engineering, University of Michigan-Dearborn. In 1998, he got another Master's degree in computer engineering, Department of Electrical and

Computer Engineering, University of Michigan-Dearborn. He has published more than 20 papers. His research interest includes image processing, machine vision and inspection, fuzzy systems and neural networks.

Brennan Hamilton received a B.S. degree in electrical engineering from GMI Engineering & Management Institute, Flint, MI, in 1990, and a M.S. degree in electrical engineering from the University of Michigan, Dearborn, in 1995.

From 1990 to 1998 Mr. Hamilton was employeed by Ford Motor Company with assignments in manufacturing and engineering. His most recent assignment was as a senior powertrain control engineer working on on-board diagnostics.

Currently, Mr. Hamilton is employeed by Siemens Automotive, North America as the marketing manager for powertrain electronics and systems integration.

Mr. Hamilton is a member of the Society of Automotive Engineers.