Intelligent Fault Diagnosis of Steer-By-Wire Automobile

Fangyuan Wu
Guangxi University of Technology, Liuzhou, China
Email: wfycool@163.com

Feng Kong and Jiangyun Yao
Guangxi University of Technology, Liuzhou, China
Email: gxkofe@163.com

***Abstract*—This paper presents an intelligent fault diagnostic approach for a steer-by-wire (SBW) system. A rough set model is utilized to reduce the redundant information. On the base of the reduction, the classifying rules can be extracted. A radical basis function (RBF) neural network optimized by particle swarm optimization (PSO) algorithm is designed to learn the fault rules that are extracted from the reduction of the redundant information. The proposed approach is simulated in MATLAB. Simulation results show that the proposed intelligent fault diagnostic algorithm provides a higher level of diagnostic accuracy than the approach without any optimization.**

***Index Terms*—fault diagnosis, steer-by-wire, rough set, radical basis function neural network, particle swarm optimization**

I. INTRODUCTION

Comfort, usability, safety, and packaging benefits of drive-by-wire systems in an automobile have been well established over many years. The challenge, however, remains in the reliability of the steer-by-wire (SBW) systems. The potentially catastrophic nature of a steering system failure requires that any practical steer-by-wire system be extremely reliable. Fault detection and diagnosis approach provides a way to improve the reliability and safety of the automobile. A great deal of research has been conducted in this area. One approach for achieving the necessary level of reliability relies on a real-time diagnostic system that can quickly and accurately diagnose a fault in the steer-by-wire system.

In this paper, the goal is designed to enhance the network training speed and diagnostic accuracy. The reduction of the fault samples is expected a simplified set for the network inputs, the effect is expected to be a higher training speed by using the reduced set. In expect of a higher diagnostic accuracy, the central position and width of the basis function in the RBF network is encoded and optimized by the PSO algorithm.

We first develop a rough set model to make a reduction for the fault information. Based on the reduction, the fault rules and be extracted. Then a neural network is utilized to learn the rules and approximate the ideal data. In the learning algorithm, the particle swarm optimization is utilized to optimize the network training speed and accuracy. Based on the optimization, the proposed algorithm was simulated in Matlab, and the simulation results show that the proposed intelligent fault diagnostic algorithm provides a higher level of diagnostic accuracy and training speed.

II. REDUCTION ALGORITHM

The rough set Ref.[1]-[4] is proposed by Pawlak Z in 1982. It deals with the classificatory analysis of data tables. The data can be acquired from measurements or from human experts. The main goal of the rough set analysis is to synthesize approximation of concepts from the acquired data. We show that first in the traditional approach and later how it evolves towards \information granules" under tolerance relation. The purpose of developing such definitions may be twofold. In some instances, the aim may be to gain insight into the problem at hand by analyzing the constructed model. The structure of the model is itself of interest. In other applications, the transparency and explainability features of the model may be of secondary importance and the main objective is to construct a classifier that classifies unseen objects well. A logical calculus on approximate notions is equally important. It is based on the concept of being a part to a degree" and is known as rough mereology.

The overall modeling process typically consists of a sequence of several sub-steps that all require various degrees of tuning and fine-adjustments. In order to perform these functions, an environment to interactively manage and process data is required. An important feature of rough sets is that the theory is followed by practical implementations of toolkits that support interactive model development. Several software systems based on rough sets exist.

It is an indeterminate theory of the data analyze, the significant characteristic of this theory is do not need any prior knowledge in data analyze. The main ideal of rough set is to delete the redundant information on the premise of keeping the basic ability of classifying. The entire research object is called the universe in rough set, and the universe is divided into several mutually-exclusive equivalent classes by the equivalent relation, which could describe any basic information of the set in the universe.

doi:10.4304/jcp.7.5.1204-1211

In rough set, a data set is represented as a table, where each row represents a case, an event, a patient, or simply an object. Every column represents an attribute that can be measured for each object; the attribute may be also supplied by a human expert or user. This table is called an information system. In many applications there is an outcome of classification that is known. This a posteriori knowledge is expressed by one distinguished attribute called decision attribute; the process is known as supervised learning. Information systems of this kind are called decision systems.

In the previous section we introduced one natural dimension of reducing data which is to identify equivalence classes, objects that are indiscernible using the available attributes. Savings are to be made since only one element of the equivalence class is needed to represent the entire class. The other dimension in reduction is to keep only those attributes that preserve the indiscernibility relation and, consequently, set approximation. The rejected attributes are redundant since their removal cannot worsen the classification. There is usually several such subsets of attributes and those which are minimal are called reduction. Computing equivalence classes is straightforward.

An equivalence relation induces a partitioning of the universe (the set of cases in our example). These partitions can be used to build new subsets of the universe. Subsets that are most often of interest have the same value of the outcome attribute. It may happen, however, that a concept such as "Walk" cannot be defined in a crisp manner.. In other words, it is not possible to induce a crisp (precise) description of such patients from the table. It is here that the notion of rough set emerges. Although we cannot define those patients crisply, it is possible to delineate the patients that certainly have a positive outcome, the patients that certainly do not have a positive outcome and, finally, the patients that belong to a boundary between the certain cases. If this boundary is non-empty, the set is rough. These notions are formally expressed as follows.

Define U={x_1,x_2,…,x_n} is a nonempty, finite set which is called the universe. Define $R \subseteq U \times U$ is the binary equivalent relation in U, then the ordered pair (U,R) is the Pawlak approximation space. For any ordered pair of $(x, y) \in U \times U$, if $(x, y) \in R$, then x and y is called undistinguishable relation in R, Otherwise, x and y is called distinguishable relation. The equivalent relation R determines a partition of the domain U/R. The sets in U/R are called the basic sets. The sets in U are called the concepts, and (U,R) is called the knowledge base. The basic sets in empty sets or any finite sets are called definite sets, others are called undefined sets. For any $X \subseteq U$, it may not be able to describe X by using knowledge module in the knowledge base. Then X is described by the lower approximate set and the upper approximate set of the approximate space (U,R), the formulas are as follows:

$$\underline{R}(X) = \{x : [x]_R \subseteq X\} = \cup\{[x]_R : [x]_R \subseteq X\}$$

$$\overline{R}(X) = \{x : [x]_R \cap X \neq \varnothing\} = \cup\{[x]_R : [x]_R \cap X \neq \varnothing\}$$

The lower approximate of X is included in the maximum definable set of X, while the upper approximate of X is included in the minimum definable set. If the upper and lower approximate of X is equal, X is definable, otherwise X is rough.

In this paper, the fault sample reduction is done by the steps are as follows:

a) The binary relation is established and judged according to the sample data, and then the suit rough set model could be founded. In this paper, equivalence relation R is judged, and the rough set model is established.

b) To establish the discernible matrix. Let system $(U, \Delta, D = \{d\})$ is a decision system, U is the universe, Δ is the cover of U, D is the decision set, U={x_1,x_2,…,x_n}, the $n \times n$ order matrix (c_{ij}) is marked as $M(U, \Delta, D)$, and the matrix is called the discernible matrix of system (U, Δ, D), in the system $\forall x_i, x_j \in U$.

$$\text{If } d(\Delta_{x_i}) \neq d(\Delta_{x_j}):$$

$$c_{ij} = \{C \in \Delta : (C_{x_i} \prec C_{x_j}) \wedge (C_{x_j} \prec C_{x_i})\} \cup \{C_s \wedge C_t : ((C_s)_{x_i} \subset (C_s)_{x_j}) \wedge ((C_t)_{x_j} \subset (C_t)_{x_i})\}$$

$$\text{Else if } d(\Delta_{x_i}) = d(\Delta_{x_j}):$$

$$c_{ij} = \Delta .$$

c) The reducing discernible function $f(U, \Delta, D)$ which refers to a Boolean function is calculated, and the function contains m Boolean variables $\overline{C_1}, \overline{C_2}, \cdots, \overline{C_m}$ which are corresponding to m covers $C_1, C_2, \cdots, C_m$ of the decision-making system. The discernible function is defined as:

$$f(U, \Delta, D)(\overline{C_1}, \overline{C_2}, \cdots, \overline{C_m}) = \bigwedge_{i,j=1}^{n} \vee (c_{ij})$$

d) The reduction of the samples is listed by the discernible function.

III. RBF NEURAL NETWORK OPTIMIZED BY PSO

A. The PSO algorithm

The Particle Swarm Optimization Ref.[5] is an evolutionary computing technology, is proposed by Dr. Kennedy and Dr. Eberhart. The particle swarm concept originated as a simulation of a simplified social system. The original intent was to graghically simulate the graceful but unpredictable choreography of a bird flock. Initial simulations were modified to incorporate nearest-neighbor velocity matching, eliminate ancillary variables, and incorporate multidimensional search and acceleration by distance. At some point in the evolution of the algorithm, it was realized that the conceptual model was, in fact, an optimizer. Through a process of trail and error, a number of parameters extraneous to optimization were eliminated from the algorithm, resulting in the very simple original implementation.

PSO is similar to a genetic algorithm (GA) in that the system is initialized with a population of random solutions. It is unlike a GA, however, in that each potential solution is also assigned a randomized velocity,

and the potential solutions, called particles, are then "flown" through the problem space.

Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. The fitness value is also stored, and is called pbest. Another "best" value that is tracked by the global version of the particle swarm optimizer is the overall best value, and its location, obtained so far by any particle in the population. This location is called gbest.

The particle swarm optimization concept consists of , at each time step, changing the velocity (accelerating) each particle towards its pbest and gbest locations. Acceleration is weighted by a random term, with separate random numbers being generated for acceleration towards pbest and gbest locations.

There is also a local version of PSO in which, in addition to pbest, each particle keeps track of the best solution, called lbest, attained within a local topological neighborhood of particles.

The process for implementing the global version of PSO is initialize a population of particles with random positions and velocities on d dimensions in the problem space; for each particle, evaluate the desired optimization fitness function in d variables; compare particle's fitness evaluation with particle's pbest, if current value is better than pbest, then set pbest value equal to the current location in d-dimensional space; compare fitness evaluation with the population's overall previous best, if current value is better than gbest, then reset gbest to the current particle's arrary index and value; change the velocity and position of the particle according to equations; loop the step until a criterion is met, usually a sufficient good fitness or a maximum number of iterations.

Particles' velocities on each dimension are clamped to a maximum velocity V_{max}. If the sum of accelerations would cause the velocity on that dimension to exceed $V_{max,}$ which is a parameter specified by the user, then the velocity on that dimension is limited to $V_{max.}$

V_{max} is therefore an important parameter. It determines the resolution, or fitness, with which regions between the present position and the target position are searched. If V_{max} is too high, particles might fly past good solutions. If V_{max} is too small, on the other hand, particles may not explore sufficiently beyond locally good regions. In fact, they could become trapped in local optima, unable to move far enough to reach a better position in the problem space.

The acceleration constants c_1 and c_2 in equation (1) represent the weighting of the stochastic acceleration terms that pull each particle toward pbest and gbest positions. Thus, adjustment of these constants changes the amount of "tension" in the system. Low values allow particles to roam far from target regions before being tugged back, while high values result in abrupt movement toward, or past, target regions.

Early experience with particle swarm optimization led us to set the acceleration constants c_1 and c_2 each equal to 2.0 for almost all applications. V_{max} was thus the only parameter we routinely adjusted, and we often set it at about 10-20% of the dynamic range of the variable on each dimension.

Based, among other things, on findings from social simulations, it was decided to design a "local" version of the particle swarm. In this version, particles have information only of their own and their neighbors' bests, rather than that of the entire group. Instead of moving toward a kind of stochastic average of pbest and gbest, particles move toward points defined by pbest and "lbest", which is the index of the particle with the best evaluation in the particle's neighborhood.

The population size selected was problem-dependent. Population sizes of 20-50 were probably most common. It was learned early on that smaller populations than were common for other evolutionary algorithms were optimal for PSO in terms of minimizing the total number of evaluations needed to obtain a sufficient solution.

The maximum velocity V_{max} serves as a constraint to control the global exploration ability of a particle swarm. As stated earlier, a larger V_{max} facilitates global exploration, while a smaller V_{max} encourages local exploitation. The concept of an inertia weight was developed to better control exploration and exploitation. The motivation was to be able to eliminate the need for $V_{max.}$

The use of inertia weight w has provided improved performance in a number of applications. As originally developed, w often is decreased linearly from about 0.9 to 0.4 during a run. Suitable selection of the inertia weight provides a balance between global and local exploration and exploitation, and results in fewer iterations on average to find a sufficiently optimal solution.

After some experience with the inertia weight, it was found that although the maximum velocity factor V_{max} couldn't always be eliminated, the particle swarm algorithm works well if V_{max} is set to the value of the dynamic range of each variable. Thus, the need to think about how to set V_{max} each time the particle swarm algorithm is used is eliminated.

Generally speaking, particle swarm optimization, like the other evolutionary computation algorithms, can be applied to solve most optimization problems that can be converted to optimization problems. Among the application areas with the most potential are system design, multi-objective optimization, classification, pattern recognition, biological system modeling, scheduling, signal processing, robotic applications, decision making, simulation and identification.

In this paper, the D dimension searching space is assumed, the total particles is n, the i particle in D dimension space is expressed as $x_i=(x_{i1},x_{i2},\ldots,x_{iD})$, the speed is expressed as $v_i=(v_{i1},v_{i2},\ldots,v_{iD})$. Each particle has the fitness decided by the optimized objective function, the best position of the particle Pbest and current position x_i is known that discovered so far. The best position of the group Gbest is known by every particle in the group, the particle position is replaced according to the following equation:

$$v_{ij}^{k+1} = w \times v_{ij}^{k} + c_1 r_1 \times (p_{ij} - x_{ij}^{k}) + c_2 r_2 \times (p_{gj} - x_{ij}^{k}) \quad (1)$$

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1} \quad (2)$$

Where v_{ij}^{k} is the speed of particle i in k iterations and j dimension component; x_{ij}^{k} is the position of particle i in k iterations and j dimension component.

B. The RBF Neural Network optimized by PSO

A radial basis function network is a neural network approach by viewing the design as a approximation problem in a high dimension space. Learning is equivalent to finding a multidimensional function that provides a best fit to the training data, with the criterion for "best fit" being measure in some statistical sense. Correspondingly, regularization is equivalent to the use of this multidimensional surface to interpolate the test data. This viewpoint is the real motivation behind the RBF method in the sense that it draws upon research work on traditional strict interpolations in a multidimensional space. In a neural network, the hidden units form a set of "functions" that compose a random "basis" for the input patterns. These functions are called radial basis function.

Radial basis functions were first introduced by Powell to solve the real multivariate interpolation problem. This problem is currently one of the principal fields of research in numerical analysis. In the field of neural networks, radial basis functions were first used by Broomhead and Lowe. Other major contributions to the theory, design, and applications of RBFNs can be found in papers by Moody and Darken.

Radial basis functions are mean to approximate multivariable functions by linear combinations of terms based on a single univariate function. This is radialised so that it can be used in more than one dimension. They are usually applied to approximate functions or data which are only known at a finite number of points, so that then evaluations of the approximating function can take place often and efficiently.

Primarily in computational applications, functions of many variables often need to be approximated by other functions that are better understood or more readily evaluated. This may be for the purpose of displaying them frequently on a computer screen for instance, so computer graphics are a field of practical use. Radial basis functions are one efficient, frequently used way to do this. Further applications include the important fields of neural networks and learning theory. Since they are radically symmetric functions which are shifted by points in multidimensional Euclidean space and then linearly combined, they form data-dependent approximation spaces. This data-dependence makes the spaces so formed suitable for providing approximations to large classes of given functions. It also opens the door to existence and uniqueness results for interpolating scattered data by radial basis functions in very general settings.

Indeed, one of the greatest advantages of this method lies in its applicability in almost any dimension because there are generally little restrictions on the way the data are prescribed. Examples below include positive definite kernels where there are no restrictions on the data except that they need to be at distinct points. This should be contrasted to, e.g., multivariable polynomial interpolation or splines. A further advantage is their high accuracy or fast convergence to the approximated target function in many cases when data become dense.

For applications it is indeed desirable that there are few conditions on the geometry or the directions in which the data points have to be placed in space. No triangulations of the data points or the like are required for radial basis function algorithms, whereas for instance finite element or multivariate spline methods normally need triangulations. In fact, the advance structuring of the data that some other approximation schemes depend on can be prohibitively expensive to compute in applications, especially in more than two dimensions. Therefore our approximations here are considered as mesh free approximations, also for instance to be used to facilitate the numerical solution of partial differential equations. On the other hand, as will be mentioned below, advanced numerical methods for computing the radial basis function approximations are required when the data set is large while more standard software is required for instance for finite elements.

The framework for the generalized radial-basis function (RBF) network shown in Figure 2 is provided by

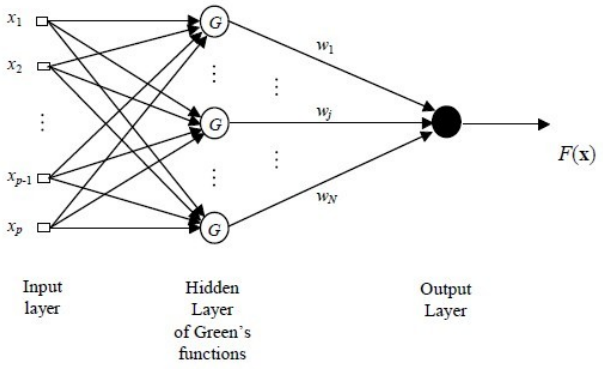


Figure.1 Regularization network

the solution to the approximation problem defined in the equation. In this network, a bias is applied to the output unit. In order to do that, one of the linear weights in the output layer of the network is set equal to a bias and the associated radial basis function is treated as a constant equal to +1.

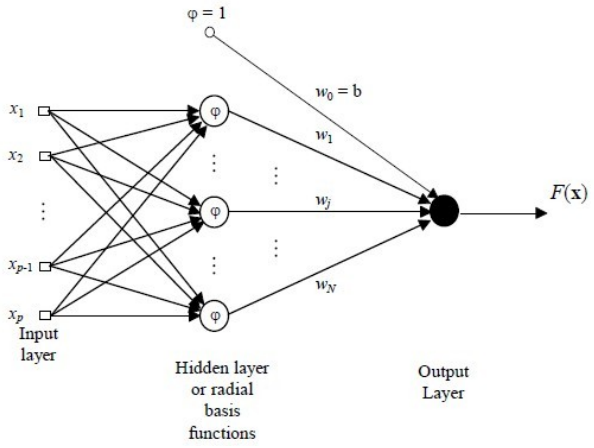


Figure2. Generalized radial basis function network

The number of nodes in the hidden layer of the generalized RBF network of Figure 2 is M, where M is ordinarily smaller that the number N of examples available for training. On the other hand, the number of hidden nodes in the regularization RBF network of Figure 1 is exactly N. In the generalized RBF network of Figure 2, the linear weights associated with the output layer, and the positions of the centers of the radial basis functions and the norm weighting matrix associated with the hidden layer, are all unknown parameters that have to be learned. On the other hand, the activation functions of the hidden layer in the regularization RBF network of Figure 1 are known, being defined by a set of Green's functions centered at the training data points; the linear weights of the output layer are the only unknown parameters of the network.

The basic ideal of the RBF neural network is: the hidden space is formed by the base of the radical basis function, and the input vector is mapped to the hidden space. The mapping relationship is established when the center of RBF network is determined. The mapping between the hidden space and output space is linear, and then the output of the network is the sum of the hidden output. Generally speaking, the mapping between the input and output is non-linear, but the output of the network is linear to the adjustable parameter.

The output equation is defined by the structure of the RBF neural network Ref.[10]:

$$Y_j = \sum_{i=1}^{q} w_{ij} \exp\left(-\frac{1}{2\sigma^2} \left\| x_k - c_i \right\|^2 \right),\ j = 1, 2, \cdots, n \qquad (3)$$

Where x_k is the input data of k, c is the center of the radical basis function, σ is the width of the basis function, w_{ij} is the connection weights between the hidden layer and the output layer, Y_j is the actual output of the j node that corresponding the input data.

Predictably, the parameters to be solved in the RBF network are as follows: the center of the basis function c 、 the width σ and the weights w. The clustering algorithm Ref.[11] is utilized to find the number of the centers, then the centers and widths are encoded by the PSO. The number of the particle is supposed to be p, and the centers is q, then the dimension of the particle is D=2q. The codes are shown in Table I.

TABLE II.
ENCODE BY PARTICLE SWARM OPTIMIZATION

Encode of the center	Encode of the width
$c_{11}, c_{12}, \cdots, c_{1q}$	$\sigma_{11}\sigma_{12}, \cdots, \sigma_{1q}$
$c_{21}, c_{22}, \cdots, c_{2q}$	$\sigma_{21}\sigma_{22}, \cdots, \sigma_{2q}$
• ••••••••••••••••••••••••••••••••	
$c_{p1}, c_{p2}, \cdots, c_{pq}$	$\sigma_{p1}\sigma_{p2}, \cdots, \sigma_{pq}$

The optimal position is searched in the $p \times 2q$ solution space by the particle group, the best position of the whole group are corresponded to the center and width of the RBF network. The fitness function of the particle is determined by the mean square error (MSE) of the parameters. Equation (4) is utilized to calculate the fitness of the particle.

$$F_{fit} = \sum_{i=1}^{p} (Y_i - T_i)^2 / 2p \qquad (4)$$

TABLE I.
FAULT DECISION TABLE

	Attributes										Decision
No.	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	D
1	1	1	0	0	0	0	0	0	1	0	D_1
2	0	1	0	0	0	0	0	1	1	0	D_1
3	0	1	1	1	0	0	0	0	0	0	D_1
4	1	0	1	1	1	0	0	1	1	1	D_2
5	0	0	1	1	1	0	0	1	1	1	D_2
6	1	1	0	0	0	1	1	1	1	1	D_2
7	0	0	1	0	0	0	0	1	0	1	D_3
8	0	0	1	0	0	0	0	0	0	1	D_3
9	1	0	0	0	0	0	0	0	1	1	D_4
10	0	0	0	0	0	0	0	0	1	1	D_4
11	0	0	0	0	0	1	1	1	0	1	D_4

Where Y_i is the actual output of particle *i*, T is the expected output of particle *i*. The group best position *Gbest* is obtained when the searching is completed by the particle, moreover, the center and width of the basis function is determined. The steps are as follows:

a) The particle number is set to p, the demension of the particle is set to 2q, and the maximum iterating times G_{max}=200,the fitness、individual best fitness $F_g(i)$、individual best position Pbest(i)、the group best position Gbest and group fitness of each particle is initialized. The threshold ε of the fitness is set, the calculation is terminated when the group fitness is lesser than this threshold.

b) With the position of particle i, the fitness $F_{fit}(i)$ of current particle in the training samples is calculated. The fitness is compared with its fintness of the best position $F_g(i)$, the particle best position Pbest(i) is replaced if $F_{fit}(i) < F_g(i)$.

c) The particle fintness $F_{fit}(i)$ of particle i is compared with the group's fitness $F_{fit,}$ the group's best position is repalced if $F_{fit}(i) < F_{fit.}$

d) The particle's position and speed is replaced by the equation (1) and (2).

e) The group's best fitness F_{fit} is compared with the threshold ε, the calculation is terminated if $F_{fit} < \varepsilon$, otherwise, the steps from b) to d) is repeated until the term is satisfied.

f) The calculation is terminated, the group's best position that refer to the center and width of the basis function is obtained.

The weights between hidden layer and output layer can be obtained by the least-square method [8], the equation is as follows:

$$w = \exp\left(q\left\|x_p - c_i\right\|^2 / c_m^2\right), \text{p=1,2,…,P; i=1,2,…,q} \quad (5)$$

Where P is the total sample numbers, c_m is the maximum distance between the selected centers.

IV. THE SIMULATION AND APPLICATION

Take the common faults of SBW system as the object to analyze, the fault features and fault reasons are listed, the fault features are taken as the attributes of the rough set, the fault reasons are taken as the decisions of the rough set, then the decision table is formed. In Table □, the attributes and their value are replaced with the symbols and numbers, where C_1 is the steering drag; C_2 is the reverse force; C_3 is the corner locked; C_4 is the steering quiver; C_5 is the steering noise; C_6 is the high-speed transmission ratio; C_7 is the low-speed ratio; C_8 is the high-speed steering; C_9 is the manipulate instability; C_{10} is the steering failure. 0 and 1 is used to express the value of each attribute, 0 is normal, 1 is abnormal. Decision D_1(1 0 0 0) shows the feeling motor fault; D_2(0 1 0 0) shows the steering gear fault; D_3(0 0 1 0) shows the angle sensor fault; D_4(0 0 0 1) shows the torque sensor fault; fault decision sets $D=\{D_1,D_2,D_3,D_4\}$.

The identification matrix $M(U,\Delta,D)$ is established according to the step *b)* in rough set, then the identification function is calculated by the step *c).*

$$f(U,\Delta,D) = \wedge\{\vee(C_{ij}) : 1 \le i \le j \le 11\}$$
$$= C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge (C_5 \vee C_6 \vee C_7) \wedge C_8 \wedge C_9 \wedge C_{10}$$

Ultimately, the reduction is obtained as:

$$\mathrm{Re}\,d(\Delta) = \{\{C_1,C_2,C_3,C_4,C_5,C_8,C_9,C_{10}\},\{C_1,C_2,C_3,C_4,C_6,C_8,C_9,C_{10}\},\{C_1,C_2,C_3,C_4,C_7,C_8,C_9,C_{10}\}\}.$$

One reduction is selected as the inputs of the network from the three reductions, and the decision is the output. After the reduction, the attribute is decreased than before, that means the lower inputs for the neural network, and the training speed is expected higher.

TABLE III.
CONTRAST OF THE DATA TRAINED BY THE NETWORK

Name	Attributes	Training time	Epochs	MSE
Before reduction	C_1-C_{10}	1.437593s	8	0.0275
After reduction	C_1-C_5,C_8-C_{10}	1.064843s	8	0.0275

As is shown in Table □, the change is not obvious in epochs and mean square error (MSE), however, the difference between before and after reduction is the training time, obviously, the training time is decreased greatly after reduction. It shows that the training speed is higher after the reduction, but the reduction has no influences on the EPOCH and MSE.

For a higher accuracy, the PSO algorithm is utilized to optimize the RBF neural network. The particle swarm scale is configured to 80; the particle dimension is configured to 16; the maximum iterating times is configured to G=200; parameter $c_1=c_2=2$; self-adapting inertia ω =0.5; the maximum speed Vmax=0.3, the network optimized by the PSO is trained with the reduction data, and the test is simulated by the other 40 fault samples, the contrast is listed in Table □.

TABLE IV.
CONTRAST OF ERRORS MADE BY THE TWO ALGORITHMS

Names	MSE	Test Errors	Max errors	Accuracy
Without PSO	6.09×10^{-32}	0.01824	0.2432	98.2%
With PSO	3.46×10^{-32}	0.00405	0.0811	99.6%

As is shown in Table □, the accuracy is higher with PSO algorithm, and the error is lower. So the conclusion is that with the PSO optimization, the neural network has a better performance.

According to Fig.3 and Fig.4, without optimization by the PSO, the output node D1 has an error about 0.2432

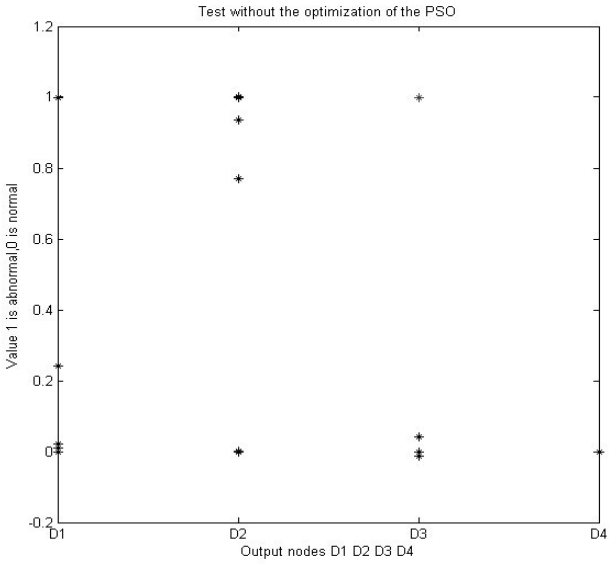


Figure.3 Test of RS-RBF network without PSO

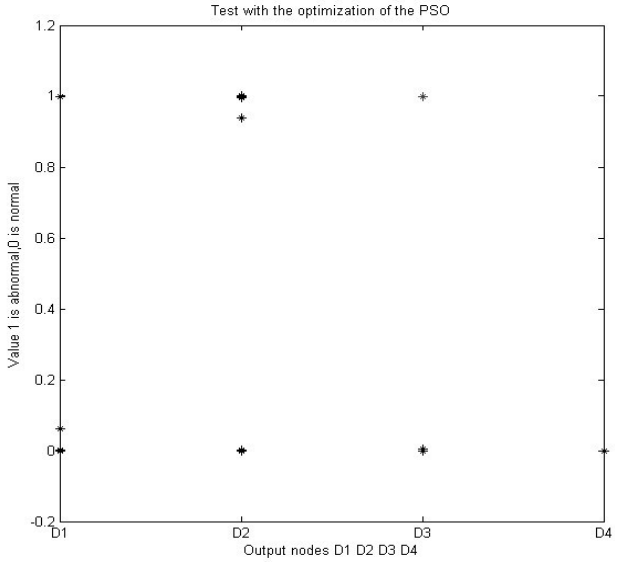


Figure.4 Test of RS-RBF network with PSO

in the output when the ideal output is 0 which presents the feeling motor is normal; the value of output node D2 is 0.7568 while the ideal output is 1 which presents the steering gear fault. The errors are decreased obviously when the optimization by the PSO is done, it is illustrated that the mean square error function (MSE) taken as the particle fitness function is effected, it has a higher diagnostic accuracy after the optimization.

II. CONCLUSION

The rough set is utilized to deal with the input of the neural network in this paper, and the RBF neural network is encoded and optimized by the PSO algorithm, then the simulation is done to proof the effective of the method. According to the contrast, the conclusion is that the dimension of the input is lower and the training speed is higher after the reduction by the rough set, and the diagnostic accuracy is higher when the center and width of the RBF neural network is optimized by the PSO algorithm.

In this paper, we introduced the rough set concept in the first section, and the reduction algorithm steps are given; in the next section, we have introduced the PSO algorithm and the RBF neural network, the optimization algorithm is given in this section; then the proposed algorithm is simulated in Matlab, the result shows that the proposed approach provides a higher level of diagnostic accuracy than the approach without any optimization.

ACKNOWLEDGMENT

This work was supported by Innovation Project of Guangxi Graduate Education (2011105940811M01).

I would also like to extend my sincere gratitude to my supervisor, Prof. Kong, for his instructive advice and useful suggestions on my thesis.

REFERENCES

[1] PAWLAK Z . Rough sets [J] . International Joumal of Computer and Information Sciences, 1982, 11(5):341-356.

[2] BASZCZYNSKI J, GRECO S, SOWINSKI R, eta1. Monotonic variable consistency Rough set approaches [J]. International Journal of Approximate Reasoning, 2009, 50(7):979-999.

[3] PAWLAK Z, ANDRZEJ SKOWRON. Rudiment of rough sets [J]. Information Sciences, 2007, 177:23-27

[4] KUANG YU HUANG, JANE C J. A hybrid model for stock market forecasting and portfolio selection based on ARX, grey system and RS theories [J]. Expert Systems with Applications, 2009, 36: 5387-5392.

[5] Kennedy J, Eberhart R C. Particle Swarm Optimization [C]. Proc of the IEEE International Conference on Neural Network. Perth, Australia, 1995:1942-1948

[6] Yang Hai-long, Sun Jian-guo. Application of PSO-based rough set theory and neural network to aeroengine fault diagnosis [J]. Journal of Aerospace Power,2009,24(2):458-464

[7] Liang Wu-ke, Zhao Dao-li, Ma Wei. Fault diagnosis of hydroelectric units based on rough set and RBF network [J]. Chinese Journal of Scientific Instrument,2007,28(10):1086-1089

[8] Zhang Ding-xue, Guan Zhi-hong, Liu Zhi-xin. RBF Neural Network Algorithm Based on PSO Algorithm and Its Application [J].Computer engineering and applications,2006,20(03):13-15

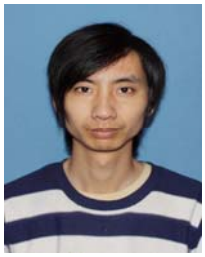
[9] Zhou Xin, Liu Zhou-hui, Analyze and research of steer-by-wire automobile [J]. Beijing automotive engineering, 2009,1(4):1-3

[10] Yamasuhara,K. Chakraborty,G. Mabuchi,H. An Efficient Method to Set RBF Network Paramters Based on SOM Training [J]. Computer and Information Technology (CIT), 2010,10(99):426-431.

[11] Zhu Jing-dong, Zhong Yong, Study of optimized RBF network in feature selection [J]. Microcomputer and its applications, 2009,28(15):76-79

[12] Liu Xin-cao, Yan Hong-wen, A RBF Neural Network Learning Algorithm Based on Improved PSO [J]. Computer thechnology and development, 2006,16(2):185-187

[13] H. Xiang-Hua, Sensor Fault Diagnosis and Reconstruction of Engine Control System Based on Auto-associative Neural Network [J]. Chinese Journal of Aeronautics. 2003, 17(1):23-37.



Fangyuan Wu received the B.E. degree in Automation from Huazhong University of Science and Technology, Wuhan, China, in 2007.

In the year 2009, he entered Guangxi University of Technology, as a graduate, where he was responsible for developing advanced control systems for automobile. He was a project host of the Innovation Project of Guangxi Graduate Education; he led projects in the area of control system development for drive-by-wire systems.

Feng Kong received the B.E. degree in Automation form Huazhong University of Science and Technology, Wuhan, China, in 1980.

He is a senior visiting scholar in Ireland; he is a holder of 8 projects and 12 patents.

Jiangyun Yao received the B.E degree in Automation from School of Yantai Nanshan University, Shandong, China.