Journal of Information Hiding and Multimedia Signal Processing ©2014 ISSN 2073-4212
Ubiquitous International Volume **5**, Number **3**, July **2014**

A Compact K Nearest Neighbor Classification for Power Plant Fault Diagnosis

Xiao-Xia Wang and Liang-Yu Ma

School of Control and Computer Engineering
North China Electric Power University, Baoding 071000, China
wtwxx@163.com

Received April, 2013; revised February, 2014

Abstract. *K nearest neighbor classification is one of the most successfully used methods in pattern recognition. Despite its simplicity and effectiveness, it suffers from several shortcomings referring to high computational cost, high storage requirement and sensitivity to noise. Prototype generation is an appropriate data reduction process to alleviate these problems. This paper employs particle swarm optimization algorithm to generate a minimal set of prototypes to correctly represent an original training set in order to improve the classification performance. After constructing an optimal set of prototypes for each of the possible classes, the compact K nearest neighbor classifier is utilized to classify by using the set of prototypes as reference. The performance of the proposed approach is evaluated on several benchmark UCI datasets. Typical faults of the high-pressure feedwater heater system are simulated under two different operating points on a full-scope simulator of a 600-MW coal-fired power unit and the results demonstrate the validity of the proposed approach.*
Keywords: K nearest neighbor classification, particle swarm optimization, power plant, fault diagnosis

1. **Introduction.** The K nearest neighbor rule (KNN) is one of the most popular supervised classifier in data mining and pattern recognition tasks. It provides a simple and intuitive method for solving a great variety of real-world classification problems and is considered one of the top ten methods in data mining [1].

For a classification problem, an original data set usually splits into a training set TR which consists of N_{TR} samples and a test set TS composed of N_{TS} samples. Each sample is represented by an M-dimensional feature vector. Let $\mathbf{x}_p = (x_{p1}, x_{p2,} \cdots, x_{pM})^T$ be a training sample from TR, and $\mathbf{x}_q = (x_{q1}, x_{q2,} \cdots, x_{qM})^T$ be a test sample from TS. We seek the classification function which is a mapping from a sample to a class ω_l, for $l = 1, 2, \cdots, C$ with C being the total number of classes. During the training process, each training sample with a known class ω is used to train the classifier, while the class of each test sample is assigned during the test process. For the KNN classifier, a test sample $\mathbf{x}_q$ is classified according to the classes of the k nearest training samples and a voting mechanism. Different distance metrics are optional to measure the similarity between the test sample and each of the training samples, and among them Euclidean distance is the most commonly used due to its simplicity. For convenience in calculation, the squared Euclidean distance is adopted and defined as

$$distance(\mathbf{x}_q, \mathbf{x}_p) = \sum_{j=1}^{M} (\mathbf{x}_{qj} - \mathbf{x}_{pj})^2 \tag{1}$$

The simplicity of the KNN classifier makes it easy to implement. However, it is necessary to determine parameter K (number of nearest neighbors), calculate the distances between the unknown sample and all the training samples, sort the distances and determine the nearest neighbors, as well as determine the categories of the nearest neighbors [2]. Therefore, KNN suffers from several drawbacks such as high computational cost, high storage requirement and sensitivity to noise, which can affect its performance.

Several approaches have been suggested and studied in order to tackle the aforementioned drawbacks. Data reduction is very useful in data mining to improve and simplify the classification model. This approach tries to find out a sufficient small set of prototypes to correctly represent the original training set with similar or even higher classification accuracy for new reduced data set. Data reduction can be divided into two different approaches: prototype selection [3, 4, 5] and prototype generation [6, 7, 8]. The former approach selects an appropriate subset of the original training data, while the latter constructs a new set of artificial prototypes with lower size and higher classification accuracy. Prototype generating is effective for improving the classification performance of KNN classifier and reducing the storage and computation requirement. Developing schemes for generating prototypes, however, has proved to be a difficult problem [9]. In this paper, we shall apply particle swarm optimization (PSO) algorithm to generate an optimal set of prototypes for all classes in order to maximize the diagnostic accuracy while minimizing the size of the prototype set.

PSO algorithm is a biologically-inspired optimization technique proposed by Kennedy and Eberhart in 1995 [10], which was motivated by the social behavior of animals, such as bird flocking and fish schooling. Each particle in the swarm adjusts its flying direction according to its own experience and other particle's experiences to search for the global optimum in the solution space. Some variants of the PSO algorithm [11, 12, 13, 14] have been studies which address the topics of convergence, parameter selection and trajectory analysis. Owing to the simple model and easiness to be implemented, PSO has been successfully applied to many fields. Some work has already been done concerning PSO algorithm in classification problems. PSO algorithm is used to extract induction rules to classify data [15], and the binary version of PSO algorithm is also used to encode induction rules [16]. Furthermore, PSO algorithm is used to generate prototypes from the training samples [8, 17].

Based on the PSO algorithm, this paper propose a novel prototype learning approach for the compact KNN classifier (CKNN). After constructing an optimal set of prototypes for each of the possible classes, a new sample is classified based on the majority of the K nearest prototype categories. PSO algorithm is used to evolve the location of prototypes per class in the input space. The proposed approach is evaluated on several real-world benchmark problems. Finally, a high-pressure feedwater heater system of a 600-MW coal-fired power unit is taken as a target system for investigation. Several faults under two different operating points are simulated in a power plant simulator and diagnosed by CKNN. The obtained results demonstrate the validity of the proposed CKNN approach.

The remainder of this paper is organized as follows. Section 2 describes the detail of PSO algorithm for prototype generation in K nearest neighbor classifier. Experimental results on several benchmark datasets are provided in Section 3 to demonstrate the effectiveness of the proposed algorithm. Section 4 presents how the proposed approach is used for fault diagnosis in the feedwater heater system of a 600-MW coal-fired power plant and the experimental results are reported in this section. Finally, conclusions are drawn in Section 5.

2. **PSO for K nearest neighbor classifier.** In this section, we employ the PSO algorithm to find a sufficient small set of prototypes from the training samples. These prototypes of the generated set can represent accurately the training samples and be used to classify as reference using the K nearest neighbor classifier.

2.1. **Encoding Scheme.** Prototype generation can be expressed as a continuous space search problem. PSO algorithm is an attractive method to optimize the location of prototypes. A prototype is represented as a feather vector that is in the same vector space as a training sample. In PSO algorithm, particles represent complete solution to the problem, so we encode a set of candidate prototypes in each particle. Prototypes are encoded sequentially in the particle in terms of their classes, therefore the class for each prototype is defined by its position inside the particle. Particles move in the search space using both local information and neighbor information and the swarm finally converges to the global optimum solution. Fig.1 represents the structure of a single particle. Each particle consists of $N_P \times C$ vectors of dimension M , where N_P is the number of prototypes for each class, C is the number of categories and M is the number of variables for each prototype. Thus, the dimension of each particle equals to $D = M \times N_P \times C$.

2.2. **Fitness Function.** To evaluate the fitness of a particle, the objective function considering the effectiveness in classification (classification error for TR) is defined as

$$f = 1 - \frac{1}{N_{TR}} \sum_{p=1}^{N_{TR}} h(\mathbf{x}_p) \tag{2}$$

Each sample in training data set TR is assigned the class of the nearest prototype from the set of prototypes encoded in the particle. If the assigned class is the same as the true class of the sample, we think that it is correctly classified by the classifier. If a sample $\mathbf{x}_p$ is correctly classified by the classifier, $h(\mathbf{x}_p) = 1$, and 0 in other cases. The algorithm searches for the best prototypes with the aim of minimizing the diagnostic error. The process is repeated until either the convergence criterion is achieved or the performance is satisfactory.

2.3. **Algorithm Equations.** In PSO algorithm, the system initially has a swarm of random solutions. Each potential solution, called a particle, is given a random initial velocity and is flown through the problem space. These particles find the global best position by competition as well as cooperation among themselves after some iteration. Supposing the dimension of a searching space is D, the total number of particles is P, the position of the ith particle is expressed as $\mathbf{x}_i = (x_{i1}, x_{i2}, ...x_{iD})$; the velocity of the ith particle is represented as $\mathbf{v}_i = (v_{i1}, v_{i2}, ..., v_{iD})$. The best previous position (which possesses the best fitness value) of the ith particle is denoted as $\mathbf{p}_i = (p_{i1}, p_{i2}, ..., p_{iD})$, which is also called *pbest*. The index of the best *pbest* among all the particles is represented by the symbol g and the position $\mathbf{p}_g = (p_{g1}, p_{g2}, ..., p_{gD})$ is also called *gbest*. Therefore the particle updates its velocity and positions according to the following equations:

$$\mathbf{v}_i(k+1) = w\mathbf{v}_i(k) + c_1 r_1(\mathbf{p}_i(k) - \mathbf{x}_i(k)) + c_2 r_2(\mathbf{p}_g(k) - \mathbf{x}_i(k)) \tag{3}$$

$$\mathbf{x}_i(k+1) = \mathbf{x}_i(k) + \mathbf{v}_i(k+1) \tag{4}$$

where w is the inertia weight; c_1 and c_2 are the cognitive and social acceleration constants; r_1 and r_2 are uniformly distributed random numbers in the range [0,1]. Velocities of particles on each dimension can be clamped with a user defined maximum velocity $V_{\max}$, which would prevent them from exploding, thereby causing premature convergence.

The inertia weight is employed to control the impact of the previous history of velocities on the current velocity. Suitable selection of inertia weight can provide a balance between

global exploration and local exploitation abilities, and on average results in less iterations required to find the optimum. As originally developed, the inertia weight w is set according to the following equation

$$w = w_{\max} - k(w_{\max} - w_{\min})/k_{\max} \tag{5}$$

where k is the current iteration number, $k_{\max}$ is the predefined maximum iteration number, $w_{\max}$ and $w_{\min}$ are the maximal and minimal weights.

Ratnaweera et al. [12] used the time-varying acceleration coefficients strategy to effectively control the global search and converge to the global best solution in PSO algorithm. The major consideration of this modification was to avoid premature convergence in the early stages of the search and to enhance convergence to the global optimum solution during the latter stages of the search. In the work, the same strategy has been adopted for the acceleration coefficients. The modification can be mathematically represented as follows:

$$c_1(k) = c_{1i} + k(c_{1f} - c_{1i})/k_{\max} \tag{6}$$

$$c_2(k) = c_{2i} + k(c_{2f} - c_{2i})/k_{\max} \tag{7}$$

where c_{1i}, c_{1f}, c_{2i} and c_{2f} are constants.

2.4. **Algorithm Pseudocode.** The procedure of prototype generation is briefly described as follows:

Step 1. Initialize a population of particles with random positions and velocities.
Step 2. Evaluate the fitness value of each particle according to Equation (2).
Step 3. Update the optimum solution of each particle.
Step 4. Update the optimum particle of the whole swarm.
Step 5. For each particle:
  5.1 Change parameter values using Equation (5), (6) and (7).
  5.2 Calculate the new velocity of the particle according to Equation (3).
  5.3 Calculate the new position of the particle according to Equation (4).
  5.4 Evaluate the fitness value of the particle according to Equation (2).
  5.5 Update the optimum solution of each particle.
  5.6 Update the optimum particle of the whole swarm.
Step 6. If a sufficiently good fitness or a maximum number of iterations is attained, go to step 7. Otherwise, go to Step 5.
Step 7. Output the best particle of the whole swarm and the algorithm terminates.

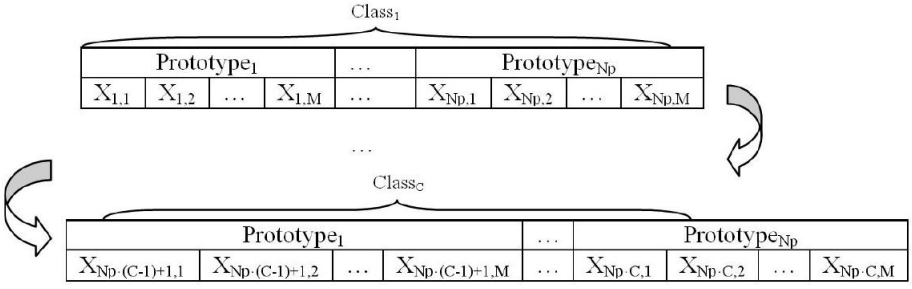

FIGURE 1. Encoding of a particle

3. **Benchmark Problems.** In this section we investigate the performance of the proposed approach and compare it with several other classification methods on five benchmark datasets. They are well-known real world classification problems taken from the UCI machine learning data repository [18]. A summary of the characteristics of these datasets (number of attributes (#Attr.), number of samples (#Samp.), and number of classes (#Clas.)) is reported in Table 1. As suggested by many classification approaches, all the datasets have been rescaled to [0,1] by a linear function. The datasets considered are partitioned using a twofold cross-validation and the results have been averaged over 10 experiments.

Table 2 compares the results of CKNN against those of nearest neighbor classifier (1NN), learning quantization vector (LVQ) [19], evolutionary nearest prototype classifier (ENPC) [20], GA for prototype selection [21] and radial basis function neural network (RBFNN) in terms of the average classification error after a fixed number of iterations in this investigation. In the experiments, the values of w_{max}, w_{min}, c_{1i}, c_{1f}, c_{2i} and c_{2f} were set to 0.9, 0.4, 2.5, 0.25, 0.25 and 2.5, respectively, according to the suggestion in [12]. The population size was set to 30, the maximum number of iterations to 600. The parameter K for the CKNN classifier was manually tuned for different datasets. The results from the Table 2 show that CKNN obtains the lower classification error than ENPC except for the Glass dataset, where the difference is not significant. Compared to 1NN, LVQ, GA and BPNN, CKNN obtains the best results for all data sets. This means that the patterns are correctly represented by the prototypes in the particle.

Table 1. Characteristics of the datasets used in the experiments

Dataset	#Samp.	#Attr.	#Clas.
Diabetes	768	8	2
Glass	214	9	6
Heart	270	13	2
Iris	150	4	3
Wisconsin	683	9	2

Table 2. Comparison with other classifiers in term of the average classification error

Dataset	CKNN	1NN	LVQ	ENPC	GA	RBFNN
Diabetes	25.13	29.69	27.86	26.56	26.30	25.78
Glass	18.22	28.04	36.92	17.76	37.78	34.58
Heart	15.93	20.74	17.04	16.30	21.48	16.67
Iris	4.00	4.67	5.34	4.67	5.34	4.00
Wisconsin	3.81	5.28	4.40	4.11	4.69	3.96

4. **Fault Diagnosis Simulation.** In this section, the capability of the proposed approach is tested with respect to the classification of faults in the feedwater heater system of a 600-MW coal-fired power plant. The diagnostic problem considered is the early identification of a predefined set of faults in the plant. The corresponding faults have been simulated under two different operating points on a full-scope simulator of a 600-MW coal-fired power unit. The power unit is running in "turbine-base boiler-following" coordination mode for the 600-MW loading condition. The automatic control systems,

including fuel control, feedwater control, high-pressure heaters' water level control, superheater steam and reheater steam temperature control, etc., are all in normal operation.

4.1. **System Description and Preprocessing.** The feedwater heater system under investigation is shown in Figure 2. The three heaters are named No. 1, No. 2, and No. 3 high-pressure heaters, according to the high-to-low sequence of their extraction steam pressure. Each heater includes three heat transfer stages: overheated steam cooling stage, saturated steam condensing stage, and drain water cooling stage.

The measured points related to this system in the data acquisition system (DAS) include: speed of each feedwater pump; feedwater pressure and temperature at pump exit pipe; each heater's inlet and exit feedwater temperatures, and its drain water temperature; each heater's inlet steam pressure and temperature; each heater's water level and the openings of its normal and emergency drain valves; feedwater pressure and temperature before economizer; water level of the deaerator and the opening of its water level control valve, etc.

For the feedwater heater system in Fig. 2, besides the normality condition (F0), 7 typical faults are generalized as follows [22, 23]:

1) Feedwater leakage from pipe to shell side in each heater (F1, F2, and F3);

2) Leakage from water-intake chamber to outlet chamber in each heater (F4, F5, and F6);

3) Inlet three-way valve's bypass side not fully closed or inner leak (F7);

To diagnose aforementioned faults, 12 feature variables, those most contribute to the fault characterization, are selected. These variables are either directly taken from the distributed control system (DCS) or obtained through simple calculation with several DCS variables. The selected variables are listed as follows:

1) Terminal temperature difference (TTD) of each heater (V1, V2, and V3);

2) Drain subcooling approach (DCA) of each heater (V4, V5, and V6);

3) Feedwater temperature rise of each heater (V7, V8, and V9);

4) Total opening of the two drain valves of each heater (V10, V11, and V12);

If a fault occurs inside a thermal system, the fault-correlated variables will increase or decrease notably, while other irrelevant variables will experience small changes. If a feature variable decreases from its nominal value after a fault occurs, its corresponding symptom may take a negative value [-1, 0). On the contrary, if a variable increases from its nominal value, its symptom may take a positive value (0,+1]. If it does not change, its fault symptom will take the value 0. With this prescaling method, a fault symptom valued within [-1,1] can reflect a possible double-sided change of a feature variable under different faults [22] . With the fault symptom prescaling method described in [23], all fault samples are normalized in the interval [-1,1]. Every fault's severity degree is reasonably set to ensure that its interrelated symptoms change distinctively and the topological structure of the system maintains unchanged.

4.2. **Prototype Generation.** In this research, we set different values for the minimum number of prototypes N_p and limit the position of each particle to the range [-1,1]. Then, the algorithm is used to search for the best prototypes for each value of N_p averaged over 10 independent runs. Fault samples under 600-MW rated loading condition are selected to form the training data set for prototype generation. The data of the 12 variables were recorded with a sampling period of 1s. All faults start after 30 seconds of steady state operation. Given that the goal is early fault diagnosis, only the data of the first 200 seconds after the beginning of the faults have been considered for each fault. A sample every 2 seconds has been considered in the interval [31,230] seconds. Thus the training set contains 1600 patterns of 8 different classes (F0, F1, F2, F3, F4, F5, F6 and F7). Fitness

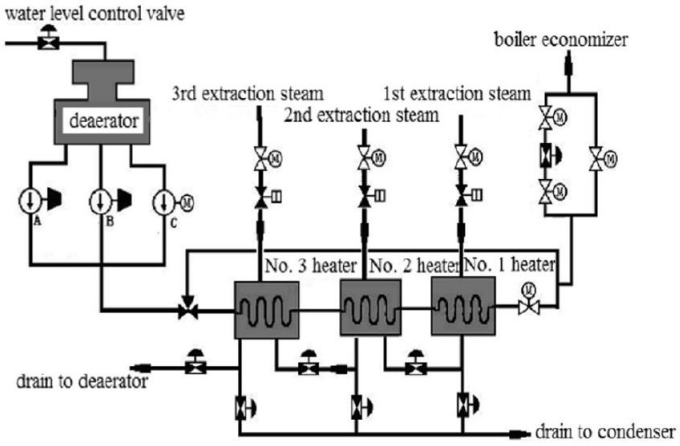

FIGURE 2. High-pressure heater system of a 600-MW unit

convergence of PSO algorithm for different values of N_p in the training procedure is given in Figure 3. With the increase of the value of N_p, the accuracy of the algorithm is not significantly improved, while the computational cost is increased sharply. So the results are compared in Figure 3 only when the values of N_p are less than 5. From the figure, it is clear that the system with 5 prototypes (N_p =5) is undoubtedly better, therefore the system with 5 prototypes of each class will be used in the following experiments.

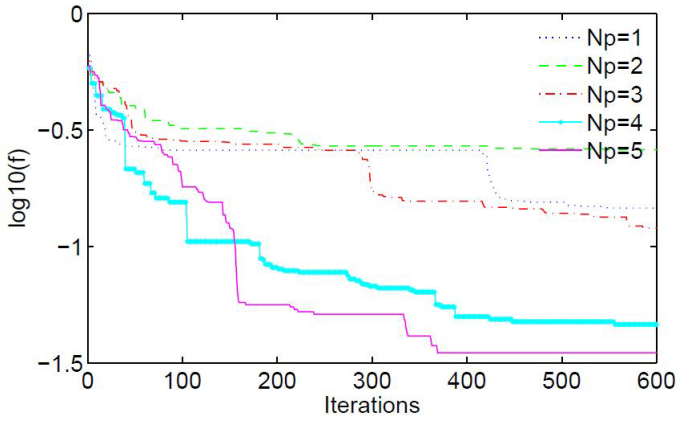

FIGURE 3. Fitness convergence for different values of N_P

4.3. **Results and Discussion.** In order to further validate the proposed approach, the KNN without prototype generation (KNN), the KNN with prototype generation randomly (RKNN) and the proposed CKNN are tested on the same data set and the diagnostic accuracies are compared. In this experiment, 7 typical faults (F1, F2, F3, F4, F5, F6, F7) were simulated and diagnosed on a full-scope power plant simulator under two different loading conditions, 600 and 480 MW. The 480-MW-load steady-state condition is obtained by dropping load with slide pressure mode from the full-load condition. The nominal reference values of the feature variables under the two operating points are taken from the simulation when the unit is operating stably with no fault. The diagnostic results under two different loading conditions using the KNN, RKNN and CKNN are given in

Table 3 and Table 4, respectively. It is clear from Table 3 and Table 4 that three methods obtain the better diagnostic results under 600-MW load point. This is mainly because the testing data under the same load condition as those of the training data are used in this experiment. Under two different loading conditions, the proposed CKNN obtains the highest diagnostic accuracies. It shows PSO algorithm can find a sufficient small set of prototypes which are generated from the samples of the training set and these generated prototypes can represent efficiently the distributions of the classes and discriminate well. RKNN selects some training samples as prototypes randomly. It means that it could select some fault-unrelated samples and there is a high degree of overlap between the prototypes. These prototypes would confuse the classification process and make diagnostic accuracies decline. Therefore, RKNN produces the worst diagnostic results. KNN uses not only the effective prototypes but also the other fault-unrelated prototypes to recognize thermal system faults, which lead to the middle classification results.

Table 3. Diagnostic results under 600-MW operating point

Fault	Diagnostic accuracy(%)		
	KNN	RKNN	CKNN
F1	96	94	100
F2	86	89	96
F3	92	90	98
F4	93	67	99
F5	94	78	100
F6	87	84	100
F7	89	86	100
Average	91	84	99

Table 4. Diagnostic results under 480-MW operating point

Fault	Diagnostic accuracy(%)		
	KNN	RKNN	CKNN
F1	92	84	96
F2	76	82	89
F3	84	88	92
F4	85	59	93
F5	82	73	95
F6	77	76	93
F7	80	75	94
Average	82	77	93

5. **Conclusions.** In this paper, a novel PSO-KNN classifier is proposed for diagnosing faults under two different operating points (600-MW and 480-MW) for power plant thermal system. Since developing schemes for generating prototypes from training patterns is a difficult problem, PSO algorithm is employed to find an ideal set of prototypes for each class with respect to a given classification problem. Five well-known benchmark problems have been used to evaluate the performance of the proposed method. Finally, the feedwater heater system of a 600-MW coal-fired power generating unit is taken as an example to further demonstrate the validity of the proposed method.

Further investigation will concentrate on diagnosing faults with varying degrees of severity and multiple faults under different operating conditions. The scope of loading conditions in fault diagnosis should be further expanded. The research needs to step into the case of gradual load-changing dynamics and multiple faults under different stable operating conditions. All of these are expected to greatly improve the practicability of the fault diagnosis system in actual power plant application.

Acknowledgment. This work is supported by the National Natural Science Foundation (Grant no. 61174111) of the People's Republic of China and the Fundamental Research Funds for the Central Universities (Grant no. 13MS89).The authors would like to express sincere appreciation to the anonymous referees for their detailed and helpful comments to improve the quality of the paper.

REFERENCES

[1] X. Wu and V. Kumar, The top ten algorithms in Data Mining. London, U.K.: chapman & Hall, 2009.

[2] L. Chuang, H. Chang, C. Tu, C.Yang, Improved binary PSO for feature selection using gene expression data, *Computational Biology and Chemistry*, vol.32, pp.29–38, 2008.

[3] D. R. Wilson, T. R. Martinez, Reduction techniques for instance-based learning algorithms, *Machine Learning*. vol.38, no.3, pp.257–286, 2000.

[4] A. Guillen, L. J. Herrera, G. Rubio, H. Pomares, A.Lendasse, I.Rojas, New method for instance or prototype selection using mutual information in time series prediction, *Neurocomputing*, vol.73, no.10-12, pp.2030–2038, 2010.

[5] S. Garcia, J. Derrac, J. Cano, F.Herrera, Prototype selection for nearest neighbor classification: taxonomy and empirical study, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol.34, no.3, pp.417–435, 2012.

[6] H. A. Fayed, S. R. Hashem, A. F. Atiya, Self-generating prototypes for pattern classification, *Pattern Recognition*, vol.40, no.5, pp.1498–1509, 2007.

[7] I. Triguero, J. Derrac, S. Garcia, F.Herrera, A taxonomy and experimental study on prototype generation for nearest neighbor classification, *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and reviews*, vol.42, no.1, pp.86–100, 2012.

[8] A. Cervantes, I. M. Galvan, P. Isasi, AMPSO: a new particle swarm method for nearest neighborhood classification, *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybern*, vol.39, no.5, pp. 1082–1091, 2009.

[9] L. Knight and S. Sen, PLEASE: A prototype learning system using genetic algorithms. I*proceedings of the 6th international conference on genetic algorithm*, San Mateo, CA, pp.429–435, 1995.

[10] J. Kennedy, R. C. Eberhart, Particle swarm optimization, *Proceedings of IEEE International Conference on Neural Networks*, pp.1942–1948, 1995.

[11] M. Clerc and J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, *IEEE Transactions on Evolutionary Computation*, vol.6, no.1, pp. 58–73, 2002.

[12] A. Ratnaweera, S. Halgamuge, H. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[13] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[14] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[15] T. Sousa, A. Silva, and A. Neves, Particle swarm based data mining algorithms for classification tasks, *Parallel Computing*, vol. 30, no. 5–6, pp. 767-783, 2004.

[16] A. Cervantes, P. Isasi, and I. Galvn, Binary particle swarm optimization in classification, *Neural Network World*, vol. 15, no. 3, pp. 229–241, 2005.

[17] L. Nanni, A. Lumini, Particle swarm optimization for prototype reduction, *Neurocomputing*, vol. 72, pp. 1092–1097, 2009.

[18] D. J., Newman, S.Hettich, C. L.Blake, C. J. Merz, UCI Repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science, 1998, URL: http://www.ics.uci.edu/ mlearn/ MLRepository.html.
[19] T. Kohonen, The self organizing map, *Proceedings of the IEEE*, vol.78, no.9, pp. 1464-1480, 1990.
[20] F.Fernandez, P.Isasi, Evolutionary design of nearest prototype classifiers, *Journal of Heuristics*, vol.10, no.4, pp.431-454, 2004.
[21] J. R.Cano, F.Herrera, M.Lozano, Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study, *IEEE Transactions on Evolutionary Computation*, vol.7, no.6, pp.561-575, 2003.
[22] L. Ma, Y. Ma, K. Y. Lee, An intelligent power plant fault diagnostics for varying degree of severity and loading conditions, *IEEE Transactions on Energy Conversion*, vol.25, no.2, pp.546–554, 2010.
[23] L. Ma and K. Y. Lee, Fuzzy neural network approach for fault diagnosis of power plant thermal system under different operating points, *IEEE Power and Energy Society General Meeting*, Pittsburgh, PA, pp.1–7, 2008.