Journal of Automation and Control, 2014, Vol. 2, No. 1, 21-32
Available online at http://pubs.sciepub.com/automation/2/1/4
© Science and Education Publishing
DOI:10.12691/automation-2-1-4

Comparing the Fault Diagnosis Performances of Single Neural Networks and Two Ensemble Neural Networks Based on the Boosting Methods

Pooria Karimi*, Hooshang Jazayeri-Rad

Department of Instrumentation and Automation, Petroleum University of Technology, Ahwaz, Iran
*Corresponding author: pooria.k@gmail.com

Received February 16, 2014; Revised February 25, 2014; Accepted February 28, 2014

Abstract This work employs potential and distinct features of artificial neural networks, in particular the feed-forward multilayer structure, to achieve fault diagnosis in chemical plants. Artificial neural networks can automatically store information by learning from historical fault data without using any qualitative or quantitative model of the system. However, different limitations are encountered in utilizing a single neural network classifier in fault diagnosis of complex large-scale chemical processes. These limitations are mostly originated from the low reliability and also high generalization error of a single neural network classifier which may be faced in real applications. An attractive remedy for these issues is to use an ensemble of different neural network models instead of relying on a single one. In this work, boosting as a general well-known approach to improve the learning performance of learning algorithms by building ensemble models is investigated. In particular, an efficient boosting method called the Stage-wise Additive Modeling using a Multi-class Exponential Loss Function (SAMMELF) is utilized. This technique can effectively overcome the main limitations of the early boosting methods for fault diagnosis applications. The potential of the SAMMELF algorithm is proved through its application to the challenging diagnosis problem of the Tennessee-Eastman Process (TEP) where the whole sets of the predefined TEP faults are considered in the classification problem.

Keywords: *artificial neural network, process fault diagnosis, adaptive boosting, Tennessee-Eastman Process*

Cite This Article: Pooria Karimi, and Hooshang Jazayeri-Rad, "Comparing the Fault Diagnosis Performances of Single Neural Networks and Two Ensemble Neural Networks Based on the Boosting Methods." *Journal of Automation and Control*, vol. 2, no. 1 (2014): 21-32. doi: 10.12691/automation-2-1-4.

1. Introduction

Changes in the physical conditions of process units, control systems or external conditions may lead to what are generally referred to as faults. Faults in the broadest sense include symptoms resulting from physical changes, such as deviations of temperature or pressure from their normal operating ranges, as well as physical changes themselves such as scaling, foaming, leaks and wear. Even changes in unmeasured process parameters such as heat or mass transfer coefficients can be considered to be faults. Generally speaking, fault is defined as any departure from an acceptable range of an observed variable or a calculated parameter of the process [1]. Faults in process equipment can give rise to off-specification production, increased operating expenses, the likelihood of line shutdown and an increasing risk of damaging environments. Hence, the growing demand for performance, efficiency, reliability and safety of industrial systems, is encouraging a growing interest both from industry and academic circles in fault diagnosis.

Process monitoring associated to fault management mission is essentially a four-stage process: fault detection, fault identification, fault diagnosis, and process recovery. (i) The task of fault detection is to determine when abnormal process conditions occur on-line. (ii) Fault identification is identifying the observation variables most relevant to diagnosing the fault. The purpose of this procedure is to focus the plant operator's and engineer's attention on the subsystems most pertinent to the diagnosis of the fault, so that the effect of the fault can be eliminated in a more efficient manner. (iii) Fault diagnosis which is the purpose of this article, is determining which fault occurred, in other words, determining the root cause of the observed out-of-control status. The fault diagnosis procedure is essential to the counteraction or elimination of the fault. (iv) Process recovery, also called intervention, is removing the effect of the fault [2].

Numerous techniques have been proposed for process fault detection and diagnosis in the past few decades. According to [3] these techniques can be broadly classified as model based and process history-based approaches. Model based approaches generally utilize results from the field of control theory and are based on parameter estimation or state estimation [4]. This approach is based on the fact that a fault will cause changes in certain physical parameters which in turn will lead to changes in some model parameters or states. It is

then possible to detect and diagnose faults by monitoring the estimated model parameters or states. When using this approach, it is essential to have the knowledge about the relationships between faults and model parameters or states. Furthermore, quite accurate models are required. The main problem with these techniques is that obtaining an exact process model is not always possible or economical in industrial systems. In addition, errors in the model can be interpreted as faults thus yielding false alarms, or can prevent faults from being detected when they occur, especially in nonlinear or uncertain systems. Hence, many process model based quantitative techniques are only applicable to linear systems with limited availability of methods for nonlinear chemical processes [5].

The process history-based methods make use of the large amount of process data obtained from recorded measured variables of the process during abnormal and normal operations. Due to the recent advances in sensing and data collection technology which provides large amounts of data collected in many processes at chemical industries, these methods are more popular than model-based approaches in real applications. Many multivariate statistical techniques for analyzing these massive datasets have been developed including Principal Component Analysis (PCA), Partial Least Squares (PLS), and Fisher Discriminant Analysis (FDA) [6]. Although most of these techniques are well designed for the fault detection, one of the most relevant techniques for the diagnosis is the supervised classification. In this technique, different operating conditions including normal and abnormal ones are treated as patterns or classes. Given that there are many datasets in the historical database, each related to a different abnormal condition (root cause); the aim is then to allocate the on-line out-of-control measurements to the most narrowly associated fault class. Classification tools are one of the most extended fault diagnosis systems in literature. They have gotten ahead because of their ease of implementation, so they do not necessitate process operators' experience or process first principles information. In addition, they can handle the inherent nonlinearity of variables existed in most chemical processes at no extra cost.

Numerous methods have been developed for supervised classification in machine learning area. These techniques which were successfully applied to fault diagnosis applications include the *k*-Nearest Neighborhood (kNN) [7], FDA [8], the Bayesian Networks (BNs) [9,10] and the Support Vector Machines (SVMs) in more recent works [11,12,13,14]. In particular, amongst the classification techniques, Artificial Neural Networks (ANNs) had an enormous attention in past years. ANNs have many useful properties concerning process fault diagnosis. They can handle nonlinear and undetermined processes where no process model is available and learn the diagnosis by means of the information of the learning data. ANNs are very noise tolerant and work well with noisy measurements. The ability to generalize the knowledge as well as the ability to adapt during their use is one of their very interesting properties. The use of ANNs in fault diagnosis is very straightforward [15]. Multi-Layer Perceptron (MLP) networks are the most popular ANNs which have been used extensively for fault diagnosis [16-21].

Despite simplicity and versatility of MLP networks, their application to process fault diagnosis in plant-wide scale is not without difficulties. One of the most important issues of employing MLP networks in process fault diagnosis is related to their over-fitting, meaning that the network shows very good performance on the training data, however behaves poorly with unseen data. Moreover, The MLP neural networks do not necessarily converge to a unique solution during the training phase and they suffer from low robustness especially when data available for training the network are not abundant. Besides, often the input-output relationship represented by a large set of fault data dispersed on a wide input feature space might not be captured adequately by a single network and so the obtained generalization performance may not be satisfactory. Hence, it is often difficult, if not impossible, to build a perfect single MLP network model for fault classification tasks in real industrial problems involving many types of faults. To encounter these issues, use of an ensemble of neural networks is proposed in literature. Ensemble or committee of neural networks is achieved through information fusion by combining different single neural network outputs by some kind of voting scheme to get a single output [22]. Using an ensemble based method may result in improving the performances of any learning system [23]. For neural networks mainly, if one takes up that the output of an individual neural network of the ensemble comprises of a true output combined with a random error constituent with zero average, then the grouping of the outputs from the individual networks lead to averaging of the random error constituents. Hence, it reduces the estimation error. Also, an ensemble of networks might reduce the over-fitting, by combining different networks with different architectures. Moreover, the risk of an unfortunate selection of a poorly performing classifier is reduced. Use of ensemble of neural networks as a general way to improve the generalization performance is quite common in neural network literature [24,25,26]. Although the advantages of ensemble network were recognized by several researchers, it is believed that no attempt except by [27] was made to apply an ensemble network for process fault diagnosis. In that work, to develop a diverse range of individual networks, each individual network is trained on a replication of the original training data generated through bootstrap resampling with replacement. The power of fusion approach in increasing neural network robustness and reliability and leading to an improved on-line diagnosis system was also demonstrated [27]. However, a quite simple chemical process comprising a Continuous Stirred Tank Reactor (CSTR) and a heat exchanger is employed to investigate the performances of different diagnosis systems.

In this work, the possibility of improvement in performance of the MLP neural network classifiers based on ensemble architecture achieved by boosting techniques is investigated through the application to the TEP fault diagnosis problem. Boosting in machine learning area refers to the general techniques attempt to improve the performance of learning methods. In particular, two powerful techniques for ensemble classification, called adaptive boosting, or AdaBoost for short, and the SAMMELF algorithm are examined in this work. Whole sets of the TEP faults are considered in this work to provide a challenging diagnosis problem. The remainder of this paper is organized as follows. Section 2 deliberates about the ensemble based learning principles and provides a brief explanation for AdaBoost, SAMMELF and MLP

neural network classification methods. The TEP fault diagnosis problem is detailed in section 3. Fault diagnosis results obtained by application to the TEP problem are presented and discussed in section 4. Finally, section 5 concludes this paper.

2. Methods

2.1. MLP Neural Network Classifier

ANNs were motivated from the examination of the human brain and consist of many basic computational elements (neurons) locally affecting other neurons through connections. Neurons or nodes in ANNs are very basic computational elements originated from their biological equivalents. A neuron can be defined by a basic function providing a conversion from *n*-dimensional space (inputs acquired from other neurons) to 1-dimensional space (an output value that it directs to other neurons). There are weights on each of the interconnections and it is these weights that are updated during the training process to ensure that the inputs produce an output which is close to its actual value, using an appropriate training rule being applied to adjust the weights. Through the training process, the network learns from examples and in doing so acquires some capabilities for generalization beyond the training data. The architecture of these models is specified by the node characteristics, network topology and learning algorithm.

The classification of process data can be carried out with the information about different classes. Then we know that certain measurement pattern correspond to normal operation and each other measurement pattern correspond to each faulty operation. The training of the neural network using this kind of information is called supervised. The MLP network is the most prevalent architecture of ANNs which can be used for supervised classification. The network contains one input layer, one or more hidden layers and one output layer. Figure 1 depicts a simple structure of the MLP network with one hidden layer where the circles represent neurons arranged in the hidden and output layers. Each connection has a weight associated with it. For input layer, the input value is forwarded straight to the hidden layer. The hidden and output layers carry out two calculations: first a weighted sum of the inputs and then the output is calculated using a non-decreasing and differentiable transfer function. Usually a sigmoidal function is used. The information propagates in one direction only through the network (feed-forward structure).

For fault diagnosis, the input pattern represents the important variables that are affected by the existing faults and the output pattern represents the fault to be identified. The input pattern is directly fed to the network input layer and the output pattern corresponds to each node in the network output layer. Then, an output node target would be either 0 (no fault) or 1 (a particular fault). Once the weights on the connections are finally adjusted in the training phase, a new vector of sensor measurements can be sent to the input nodes of the network to be classified. Very little computation time is needed for this step. The degree of misclassification that occurs is a function of how well the knowledge stored in the connections and weights.

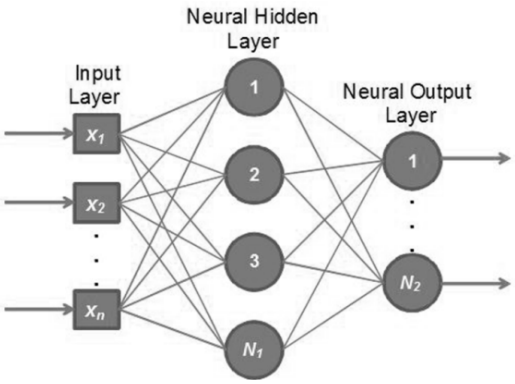

Figure 1. MLP neural network with one hidden layer

2.2. Ensemble Models for Classification

The idea of building ensembles of classifiers has gained interest in the last decade. Instead of building a single complex classifier, one aims at designing a combination of several simple (weak) classifiers (see Figure 2). For instance, instead of training a large neural network, we train several simpler neural networks and combine their individual outputs in order to produce the final output. The basic idea is to generate several classifiers and pool them in such a way as to improve on the performance of any single one. Even in some cases, even though the ensemble may not result in better than the optimum single classifier, it could reduce or remove the risk of choosing an inappropriate single classifier.

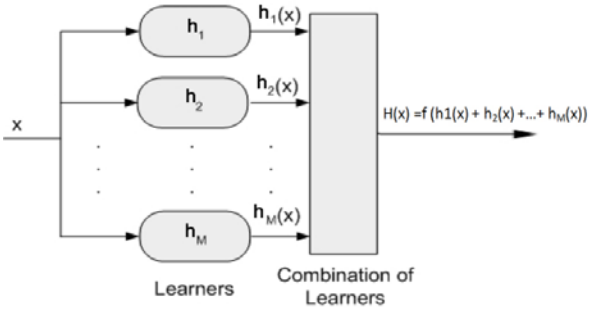

Figure 2. The concept of an ensemble of classifiers

All ensemble models are built on two key components: first, a strategy is needed to build diverse classifiers; second, a strategy, is needed to combine the outputs of the individual classifiers that make up the ensemble in such a way that the correct decisions are amplified, and the incorrect ones are cancelled out. Diversity means the base classifiers of an ensemble disagree with each other as much as possible particularly with respect to misclassified instances. Specifically, we need classifiers whose decision boundaries are sufficiently different from those of others. Various techniques have been suggested for obtaining diversity in the base models of an ensemble, e.g. using different training parameters [22], different training patterns [28], different feature subsets [29] and different learning methods for each classifier of the ensemble [30]. For combining the individual decisions, the most popular (and simplest) method is the majority voting rule, although there exist other more complex schemes (e.g., average, minority, medium and product of votes) [31]. Ensemble based classification methods have been widely studied in such diverse fields as image segmentation, data

mining from noisy data streams, credit card fraud detection, sensor networks, image/speech/handwritten recognition, fault diagnosis, etc [32].

Given: $(x_1,y_1),\ldots,(x_m,y_m)$, Where $x_i \in X$, $y_i \in Y = \{1,2,\ldots,K\}$

1. Initialize the weights $w_i^1 = 1/m$, $i=1, 2, \ldots, m$
2. For $t=1$ to T
   (a) Fit a classifier $h^t(x)$ to the training data using weights w_i^t
   (b) Compute

$$err^t = Pr_{i\sim w_i^t}[h^t(x_i) \neq y_i] = \sum_{i=1,\, h^t(x_i)\neq y_i}^{m} w_i^t \Big/ \sum_{i=1}^{m} w_i^t$$

If $err^t > 1/2$, then $t=T-1$ and abort loop.

   (c) Compute

$$\alpha^t = log\frac{1-err^t}{err^t}$$

   (d) Set

$$w_i^t \leftarrow \begin{cases} w_i^t . \exp(\alpha^t) & if \quad h^t(x_i) \neq y_i \\ w_i^t & otherwise \end{cases} \quad i = 1,2,\ldots,m$$

   (e) Renormalize w_i^t
3. Output

$$H(x) = arg\max_{y\in Y} \sum_{t=1,\, h^t(x)=y}^{T} \alpha^t$$

Figure 3. The boosting algorithm AdaBoost

2.2.1. Multi-Class Boosting

Among the different methods for constructing diverse classifiers in an ensemble classification system, most techniques are based on using different training patterns for each classifier. One of the most popular approaches for this purposes are based on boosting techniques which have considerable success in various applications [33]. Boosting encompasses a family of methods which attempts to "boost" the accuracy of any given learning algorithm. The well-known AdaBoost algorithm which is the focus of this paper is introduced in 1995 by Freund and Schapire [34] has become widespread due to its capability of solving many of the practical difficulties of the earlier boosting algorithms.

A Pseudocode for AdaBoost is given in Figure 3. The algorithm takes as input a training set $(x_1, y_1), \ldots,(x_m, y_m)$ where each x_i belongs to some domain or instance space X, and each label y_i is in some label set $Y=\{1, 2, \ldots, K\}$ where K is the number of classes. The goal is to find a classification rule $H(\boldsymbol{x})$ from the training data, so that when given a new input $\boldsymbol{x}$, we can assign it a class label y from $\{1, \ldots, K\}$. AdaBoost constructs a composite strong classifier by sequentially training weak or base classifier classifiers repeatedly in a series of rounds $t = 1, \ldots, T$. One of the main ideas of the algorithm is to maintain a distribution or set of weights over the training set. Starting with the un-weighted training sample, AdaBoost builds a classifier that produces class labels. If a training data point is misclassified, the weight of that training data point is increased (boosted) so that the weak learner is forced to focus on the hard examples in the training set. This is achieved by the rule represented in Figure 3. This rule is based on the weak classifier error rate. Notice that the error is measured with respect to the weights of the training data on which the weak learner was trained (step 2b in Figure 3). A second classifier is built using the new weights which are no longer equal. Again, misclassified training data have their weights boosted and the procedure is repeated. A score relating to the goodness of a classifier measured by its error is assigned to each classifier (α^t). The final classifier is defined as a weighted majority vote of the T weak classifiers. It should be noted that that this boosting algorithm, is sometimes denoted as adaboost.M1 in literature to emphasize on the capability of handling multi-class classification problems [33]. However, we simply refer to this algorithm as AdaBoost through this paper.

In practice, the weak learner may be an algorithm that can be trained directly based on the weights on the training examples. This method of boosting is called reweighting. However, some learning algorithms don't permit training with respect to a weighted cost function and require an un-weighted set of examples. In this case, resampling with replacement (using the probability distribution w) can be used to approximate a weighted cost function. In this case, examples with high probability occur more often than those with low probability, while some examples may not occur in the sample at all, although their probability is not zero. Neural networks can be trained directly with respect to a distribution over the learning data by weighting the cost function. However, using weighted cost function may cause numerical problems [35]. Hence, in this work, using neural network as the weak learner, the resampling boosting is utilized.

Essentially, AdaBoost has many benefits. It is fast, straightforward and easy to be implemented. It has no parameters to be tuned (except for the number of iterations). It needs no previous information about the weak learner and therefore can be easily joined to any method for finding weak hypotheses. AdaBoost has very interesting theoretical properties; in particular it can be shown that the error of the composite classifier on the training data decreases exponentially to zero as the number of combined classifier is increased provided that each classifier yields a weighted error that is less than 50% [34]. More importantly, however, bounds on the generalization error of such a system have been formulated in [36]. These are based on a notion of margin of classification, defined as the difference between the score of the correct class and the best score of a wrong class. It has been shown empirically in various applications that AdaBoost tends to be robust to over-fitting [37,38,39].

Given: $(x_1,y_1),\ldots,(x_m,y_m)$, Where $x_i \in X$, $y_i \in Y = \{1,2,\ldots,K\}$

4. Initialize the weights $w_i^1 = 1/m$, $i=1, 2, \ldots, m$
5. For $t=1$ to T
   (f) Fit a classifier $h^t(x)$ to the training data using weights w_i^t
   (g) Compute

$$err^t = Pr_{i\sim w_i^t}[h^t(x_i) \neq y_i] = \sum_{i=1,\, h^t(x_i)\neq y_i}^{m} w_i^t \Big/ \sum_{i=1}^{m} w_i^t$$

If $err^t > 1 - 1/K$, then $t=T-1$ and abort loop.

   (h) Compute

$$\alpha^t = log\frac{1-err^t}{err^t} + \mathbf{log(K-1)}$$

   (i) Set

$$w_i^t \leftarrow \begin{cases} w_i^t . \exp(\alpha^t) & if \quad h^t(x_i) \neq y_i \\ w_i^t & otherwise \end{cases} \quad i = 1,2,\ldots,m$$

   (j) Renormalize w_i^t
6. Output

$$H(x) = arg\max_{y\in Y} \sum_{t=1,\, h^t(x)=y}^{T} \alpha^t$$

Figure 4. The boosting algorithm SAMMELF

The main disadvantage of the AdaBoost is that it requires the weak classifier have prediction error less than 1/2 (with respect to the distribution on which it was

trained). The expected error of a hypothesis which randomly guesses the label is 1-1/*K*, where *K* is the number of possible labels. Thus the AdaBoost requirement for $K = 2$ is that the prediction is just slightly better than random guessing. However, when $K>2$, the requirement of AdaBoost is much stronger than that, and might be hard to meet. To solve this issue, a new modification of AdaBoost is proposed by [40] in which the new algorithm only requires the performance of each weak classifier just to be better than random guessing similar to AdaBoost in the two class case. The new algorithm is called SAMMELF. They provided a statistical justification for the algorithm by using a novel multi-class exponential loss function and the forward stage-wise additive modeling technique. The SAMMELF algorithm is presented in Figure 4. Notice that SAMMELF is very similar to AdaBoost with the only difference in weight updating rule as presented in the step 2c of the algorithm shown in Figure 4. Now in order for α^t to be positive, we only need $(1-err^t) > 1/K$, or the accuracy of each weak classifier to be better than random guessing rather than 1/2. As a consequence, the new algorithm puts more weight on the misclassified data points when compared to AdaBoost. It is worth noting that when $K = 2$, SAMMELF reduces to AdaBoost.

An experimental comparison between SAMMELF and AdaBoost methods is reported in [40] based on a simple three-class simulation example. The results are provided in Figure 5. This figure (upper row) shows how AdaBoost breaks using ten-terminal node trees as weak classifiers. As we can see (upper right panel), the test error of AdaBoost decreases for a few iterations, then levels of around 0.53. What has happened can be understood from the upper left and upper middle panels: the err^t starts below 0.5; after a few iterations, it overshoots 0.5 (α^t below 0), then quickly hinges onto 0.5. Once err^t is equal to 0.5, the weights of the training samples do not get updated ($\alpha^t = 0$), hence the same weak classifier is fitted over and over again but is not added to the existing fit, and the test error rate stays the same.

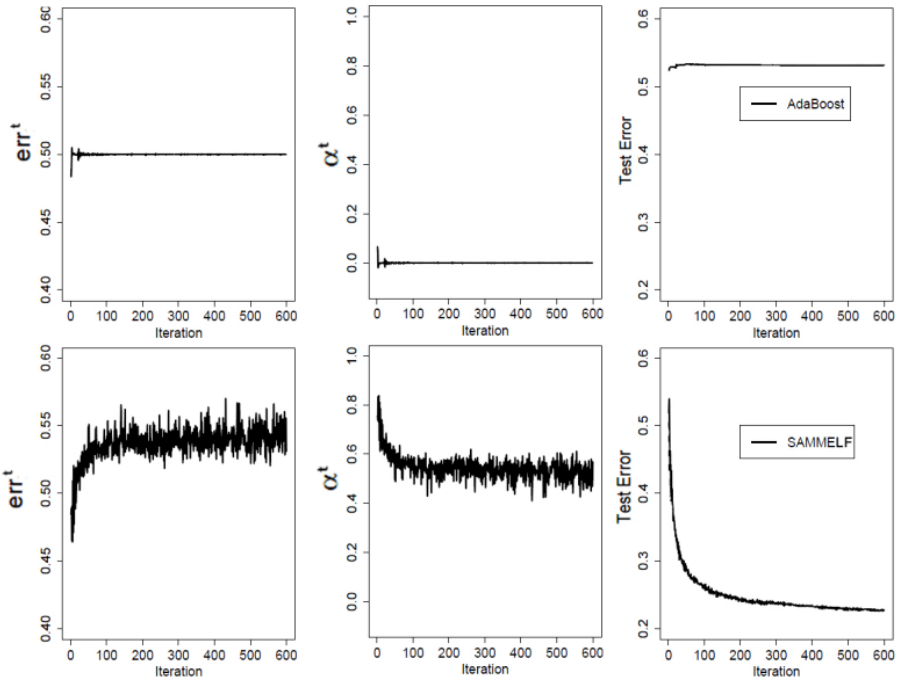

Figure 5. Comparison of AdaBoost and the new algorithm SAMMELF on a simple three-class simulation example as reported in [40]

3. Application: Tennessee-Eastman Process

The TEP problem was created by the Eastman chemical company to provide a realistic industrial process in order to evaluate process control and monitoring methods [41]. A simplified process flow sheet of TEP is plotted in Figure 6. TEP is a chemical process composed of five major operation units: a reactor, a condenser, a compressor, a stripper and a separator. Four gaseous reactants *A*, *C*, *D*, *E* and an inert *B* are fed to the reactor where the liquid products *F*, *G* and *H* are formed. The TEP represents a challenging real diagnosis problem that has been widely used for many research experiments in fault detection and control. Main diagnosis difficulties in this problem come from the number of different faults to be managed and the wide faults diversity. Random and step biases as well as soft drifts on the process variables must be managed in this complex diagnosis case study. Besides, some of the considered disturbances cause insignificant biases in the variables because of the adjusted control strategy, which makes very difficult to find out the biases source causes [12].

The TEP is unstable in open-loop. Chiang et al. [8] simulated the plant by using a closed-loop control structure adopted in [42]. Their generated simulation data which can be accessed in *http:rrbrahms.scs.uiuc.edu* were employed in this work. The process contains 41 measured variables (*XMEAS*) and 12 manipulated variables (*XMV*). However, only 52 variables are employed in this problem because an input variable (the reactor agitator speed) is constant. All the process measurements include Gaussian noise. An observation vector at a particular time instant is given by:

$$x = \left[XMEAS_1 \ldots XMEAS_{41} XMV_1 \ldots XMV_{11} \right] \quad (1)$$

The TEP simulation contains 21 preprogrammed faults. Sixteen of these faults are known and five are unknown. These faults are listed in Table 1. Two sets of training data and test data for 22 kinds of operation conditions, i.e. 21 kinds of faulty and the other one of normal status were generated. Each faulty data set started with no faults, and the faults were then introduced for the duration of 1 and 8 simulation hours into the run, respectively, for the training and testing data sets. The data were sampled every 3 min. The total number of observations generated for each run was 500 and 960 for the training and testing data sets, respectively. However, only 480 and 800 observations were correspondingly collected after the introduction of the fault.

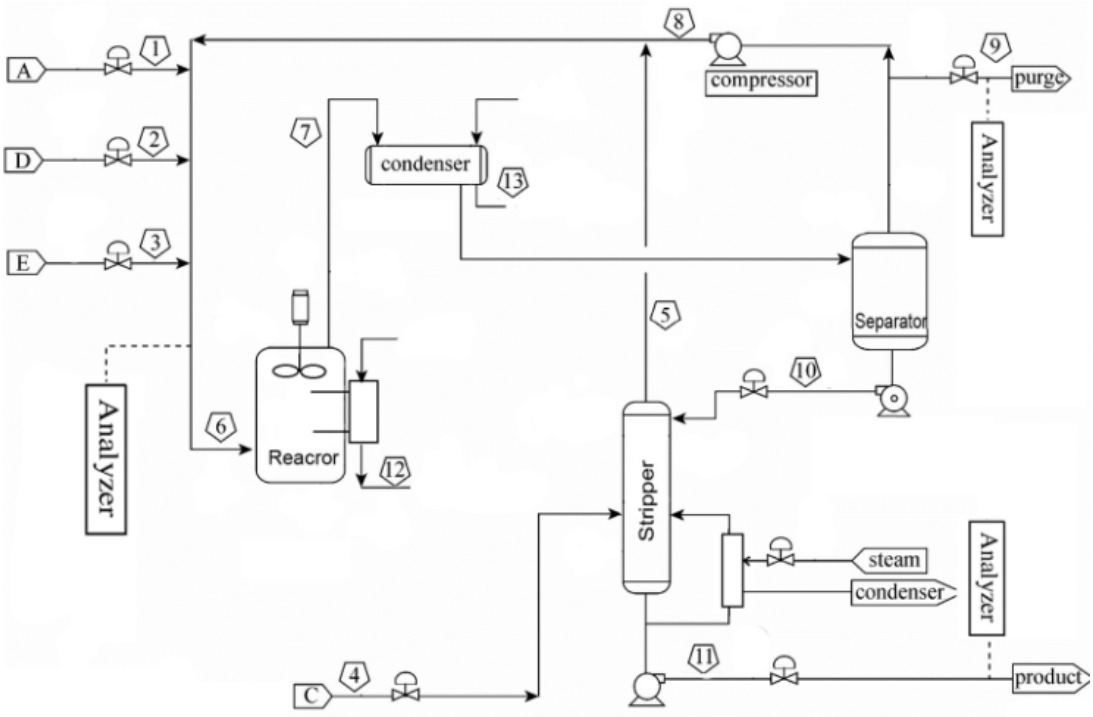

Figure 6. The process flow sheet of TEP

Table 1. Types of faults in the TE process

Fault	Description	Type
1	A/C feed ratio, B composition constant (stream 4)	Step
2	B composition, A/C ratio constant (stream 4)	Step
3	D feed temperature (stream 2)	Step
4	Reactor cooling water inlet temperature	Step
5	Condenser cooling water inlet temperature	Step
6	A feed loss (stream 1)	Step
7	C header pressure loss-reduced availability (stream 4)	Step
8	A, B, C feed composition (stream 4)	Random
9	D feed temperature (stream 2)	Random
10	C feed temperature (stream 4)	Random
11	Reactor cooling water inlet temperature	Random
12	Condenser cooling water inlet temperature	Random
13	Reaction kinetics	Slow drift
14	Reactor cooling water valve	Sticking
15	Condenser cooling water valve	Sticking
16	Unknown	Unknown
17	Unknown	Unknown
18	Unknown	Unknown
19	Unknown	Unknown
20	Unknown	Unknown
21	The valve for Stream 4 was fixed at the steady state position	Constant position

4. Results and Discussion

4.1. Single Neural Network Classifier

The raw data obtained from the TEP simulation was used to construct the different classifies discussed in this section. First, in order to get a deeper knowledge on the ANN performance, different single neural network classifiers are evaluated under different conditions. Different neural network classifiers are trained when the number of fault classes is progressively increased as shown by the fault numbering listed in Table 1. The networks used here are single hidden layer feed-forward neural networks. For each case, the number of output neurons is fixed. Each output neuron corresponds to one normal state and the matching considered fault situation. The number of hidden neurons however needs be determined. We always utilized 20 hidden neurons which experimentally found to be an optimal choice between the network training time and the classification performances in different cases. The results obtained are depicted in Figure 7 for the training and also test data. The diagnosis performance is indicated as the error rate which is simply defined as the rate of misclassified points in the whole data set. Here, misclassification indicates a point whose class is determined incorrectly. As expected, Figure 7 shows how the error rate globally increases when more classes are considered and the size of the network increases.

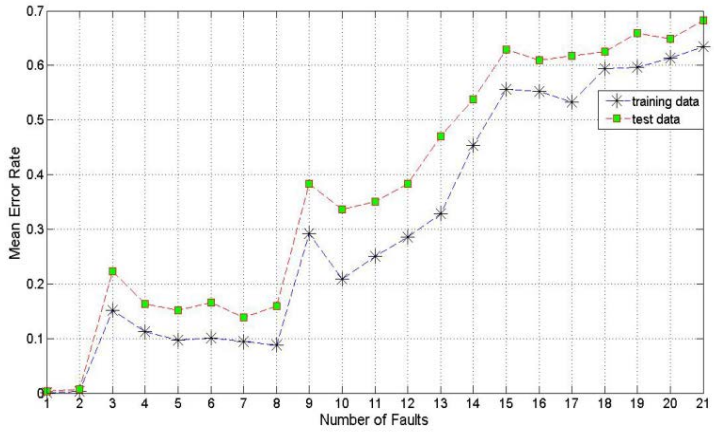

Figure 7. Error rate versus the number of considered faults. Number of faults indicate that faults 0+1+…+*n* are considered

In the employed MLP neural networks, both of the hidden and output layer neurons utilize the sigmoid activation functions. The sigmoid activation function naturally bounds the neural network outputs within the range [0,1]. In the training phase, a target value of 1 is considered for the output node corresponding to the label of the current training dataset, while the target values for all other output nodes are set to 0. For a new testing example, the labeled prediction corresponds to the node with the highest confidence. It should be noted that the Scaled Conjugate Gradient (SCG) algorithm is used in this paper for the training purpose of the MLP network [43]. In order to deal with the over-fitting issue in the training step, the early stopping technique is employed. For this purpose, the training data set is randomly divided into two subsets: the learning and the validation sets with proportion of 7 to 3, respectively. The network is trained using the learning data-set; however, its performance is monitored on the validation data set. The validation error normally decreases during the initial phase of training, as does the learning set error. However, when the network begins to over-fit the data, the error on the validation set typically begins to rise. Hence, the network training is stopped when the minimum error is reached on the validation data, i.e. when the validation error begins to rise. Otherwise, a maximum of 200 epochs of training must be utilized. Also, the entrapment in local minima problem for ANNs, sometimes necessitates the training procedure to be repeated several times each time employing different initial weights to increase the chance of selecting the best model. This procedure is considered for each case and consequently, the network with the best classification result has been chosen.

4.2. Boosting Based on AdaBoost

As mentioned before in section 2, the basic AdaBoost algorithm converges (leads to reduced training error) if the basic classifier yields a weighted error that is less than 0.5. Hence, the network trained based on the whole fault sets of the TEP problem (21 fault cases in Figure 7) cannot be directly boosted by the basic AdaBoost algorithm because it always produces error more than 0.5. Hence, we apply and evaluate the AdaBoost method to the largest network having error less than 0.5, i.e. the network which diagnoses 14 fault classes. For this purpose, first a uniform weighting distribution is assigned to the original training samples. We consider the trained network as the first round classifier of the boosting ensemble and based on its observed errors for the original training samples, the weight distribution is updated (step 2d in Figure 3). The new weighting distribution is used to generate a new training set by randomly picking samples with replacement from the current training set. The new dataset size obtained by resampling is equal to the size of the current training data set. Because the new distribution uses more weight on the training samples which were wrongly classified before, the new training set obtained by resampling on the current training set contains more difficult samples in classification view. The second network is then trained using the new training set which is more focused on the classification of the hard examples. Then all the training samples are passed through this network and the weighted error is computed and used to obtain new modification for the weighting distribution of the third round. This procedure is repeated in sequential rounds. Meanwhile, because the final networks are trained using more difficult data values than the previous ones, the weighted error rate on the training dataset normally increases. Finally the boosting procedure is stopped when the obtained weighted error exceeds the value of 0.5. In addition, in order to take into account the fact that a network may become trapped into a local minimum of the cost function in each round, the training will be restarted with new random initial weights if the current weighted training error exceeds 0.5. The maximum number of restarts for a given network is taken to be five, after which the boosting algorithm terminates. Note that, each training task is performed based on the early stopping method; i.e. the training is stopped when the network either reaches the minimum mean squared error in the validation data set or passes 200 epochs as described before.

The result obtained in this part is demonstrated in Figure 8. In this figure, the error rates on the training and test samples of the boosted classifiers, as the number of networks is increased, are presented. In each round of boosting, the outputs of current trained classifier and all other previous classifiers are combined through the majority voting method described in the step 3 of the algorithm shown in Figure 3. As can be observed in Figure 8, the training error and the test error in sequential rounds are normally decreased. This implies the general success of the boosting mission. However, after the fifth round the weighted error goes beyond 0.5, hence the

boosting procedure is aborted. The final error rate of 0.28 and 0.397 are respectively obtained for the training and testing data by the ensemble classifiers compared to the corresponding error rates of 0.453 and 0.538 generated by the single un-boosted network.

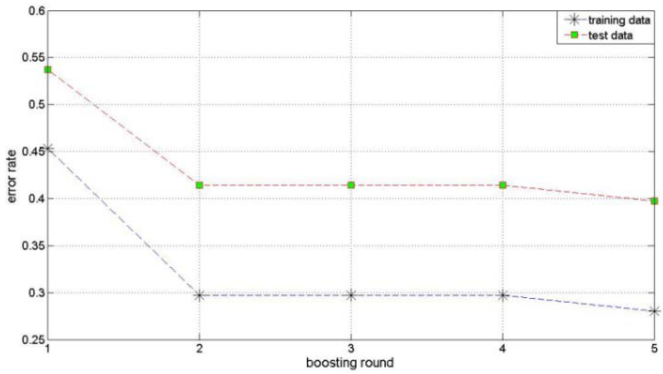

Figure 8. Boosting procedure for the network diagnoses of faults 1 to 14 of TEP obtained by AdaBoost

A remarkable feature of the ensemble networks reported in literature is their resistance to over-fitting. In other words, while the error on the training set is decreased in sequential rounds the generalization error is always bounded in the ensemble classifiers without concerning whether the single classifiers are over-fitted or not. In many experiments, it has been observed that the generalization error continues to decrease toward an apparent asymptote after the training error has reached zero. Schapire et al. [36] suggested a possible explanation for these behaviors based on the definition of margin of classification. After that, many empirical evaluations of AdaBoost also provided an analysis of the so-called margin distribution. Consider an ensemble classification scheme obtained by a majority voting combination of different single classifiers. Suppose that the weights assigned to the different base classifiers are normalized. The classification margin for each example is defined as the difference between the weight assigned to the correct label and the maximal weight assigned to any single incorrect label. It is easy to see that the margin is a number within the range [-1,1] and that an example is classified correctly if and only if its margin is positive. Moreover, the magnitude of the margin can be interpreted as a measure of confidence in the prediction. Schapire et al. [36] proved that larger margins on the training set translate into a superior upper bound on the generalization error. They also showed that that boosting is particularly aggressive at reducing the margin (in a quantifiable sense) since it concentrates on the examples with the smallest margins (either positive or negative).

Figure 9 shows the margin cumulative distributions (MCD) obtained in our experiment. MCD represents the fraction of examples in the training data whose margin is at most x as a function of $x \in [-1,1]$. The margin distribution is plotted after two rounds of boosting and also for the final obtained ensemble which is obtained after the 5th round. It is clear that the number of examples with high margin increases when more classifiers are combined by boosting. This result also confirms the success of boosting in our application.

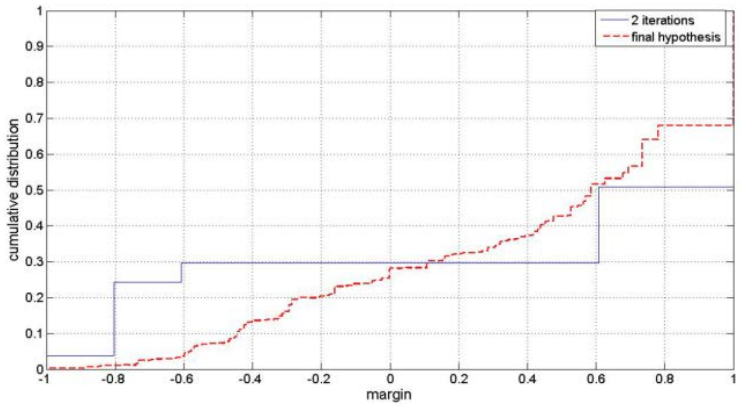

Figure 9. The margin distribution of the training examples obtained in the boosting of the network diagnoses of faults 1 to 14 of TEP by AdaBoost

4.3. Boosting Based on the SAMMELF Method

The AdaBoost algorithm cannot be employed for the boosting of the networks which classifies large sets of TEP faults. Also, for the largest possible classification problem, the AdaBoost based boosting procedure could not be executed for more than 5 rounds. In this section, we first apply the SAMMELF boosting method on the same network boosted by AdaBoost to obtain a comparative view. The SAMMELF boosting algorithm is described in Figure 4. As stated before, the SAMMELF algorithm is similar to AdaBoost. The only difference originates from the weight updating function. The one which is suggested by the SAMMELF method requires the weak learner to perform just slightly better than random guessing.

Whereas, for the AdaBoost algorithm a weighted error less than 0.5 is always required. In implementation of the SAMMELF algorithm, similar to AdaBoost, the trained network with 20 hidden layers is used as the first round classifier and based on its observed training error, the weighting distribution is updated for the next round. Also, the same procedure of network learning in each round, based on early stopping technique, is carried out again. The final classifier is obtained by the majority voting technique expressed in the step 3 of the algorithm shown in Figure 4. The consequent result is shown in Figure 10. Despite the fact that the boosting algorithm basically reduces the errors obtained on the training data, however the errors of the test data are also improved in the same manner. The resistance of the SAMMELF method to over-fitting is also exhibited in this figure. The boosting procedure can be proceeded to higher rounds, compared to AdaBoost, without being aborted due to large weighted errors. However, after 20 rounds, no improvement in training error is seen by this method. The final results of the training and generalization errors are presented in Table 2 which indicates a superior performance for the SAMMELF boosting method. The MCD curve obtained by the SAMMELF algorithm is also depicted in Figure 11 for three different rounds (iterations). Similar to the results shown in Figure 9 for the AdaBoost method, Figure 11 shows the achieved performance of the SAMMELF method in boosting the margin of the training data.

Table 2. Classification performances of different methods in diagnosis cases of faults 1 to 14 of TEP

	Single un-boosted network	Boosting by AdaBoost	Boosting by SAMMELF
Training error	0.453	0.280	0.182
Test error	0.538	0.398	0.382

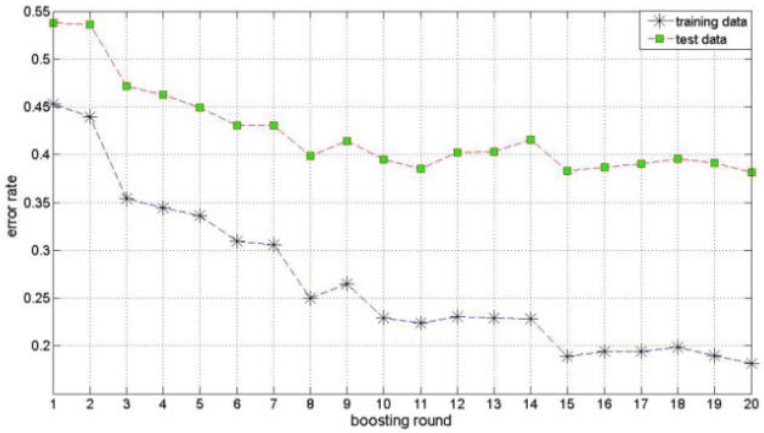

Figure 10. Boosting procedure for the network diagnoses of faults 1 to 14 of TEP obtained by SAMMELF

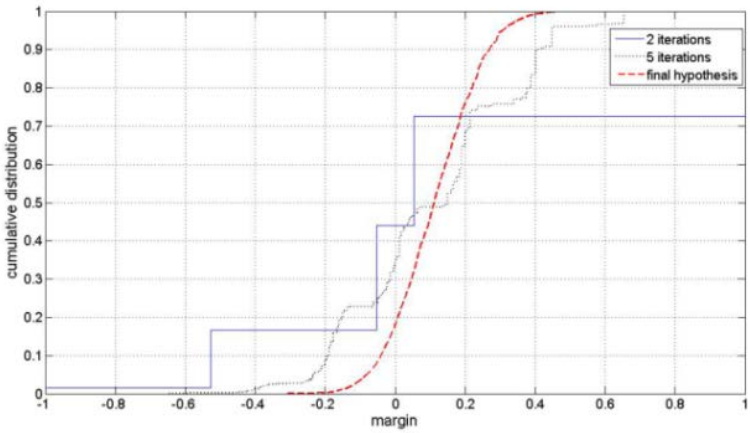

Figure 11. The margin distribution of the training examples obtained in the boosting of the network diagnoses of faults 1 to 14 of TEP by SAMMELF

We applied the SAMMELF boosting method to the largest neural networks diagnosing all the TEP faults because the generated training errors by this networks are less than the random guessing (less than 1-1/K where K is equal to 22 classes of data). The corresponding result obtained is presented in Figure 12. As seen in this figure, both the training and the generalization errors continue to decrease to their associated apparent asymptotic levels. The training and test errors are 0.349 and 0.536, respectively. The interrelated values achieved for the single network are 0.634 and 0.682. The errors for the single network were obtained after 20 iterations (no improvement was observed after the round 20).

The MCD obtained by the SAMMELF algorithm applied to the whole TEP faults is also depicted in Figure 13 for three different rounds. As a general conclusion, due to the fact that an error less than the random guessing can always be obtainable in any large classification problem, the SAMMELF method demonstrates a superior substitute for the conventional AdaBoost algorithm in real diagnosis problems.

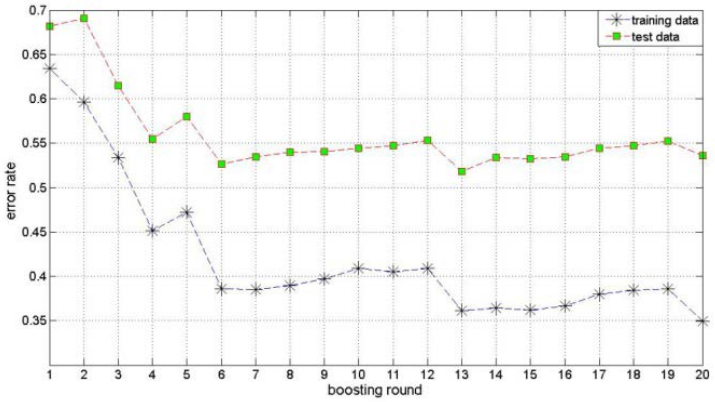

Figure 12. Boosting procedure for the network diagnoses of the whole TEP faults obtained by SAMMELF

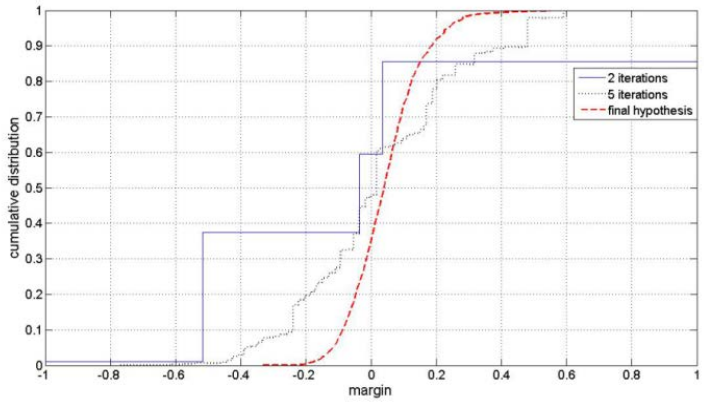

Figure 13. The margin distribution of the training examples obtained in the boosting of the network diagnoses of the whole TEP faults by SAMMELF

Using one fault pattern at a time, the single network and the boosted network obtained by the SAMMELF method are tested to determine a general view of the capabilities of these two diagnosis systems under different fault conditions. Here, the whole TEP diagnosis problem is considered. Each test pattern is applied to both of the diagnosis methods and consequently the corresponding misclassification errors are computed. The results are presented in Figure 14. As seen in this figure, some faults cannot be diagnosed by the single neural network classifiers at all. These include the challenging faults of 3, 9 and 15 which were also referred to as unobserved faults in some previous works [44]. On the other hand, the boosting improvement are more noticeable for the poorly identified faults. This reveals the fact the boosting algorithm heavily concentrates on the hard-classified examples. However, in cases such as for the faults 1, 6, 17, 18, and 20 the diagnosis performance is decreased in the boosted network. This indicates that the application of the boosting method, sacrifices the diagnosis strength of some of the well-diagnosed faults to attain a minimum level of diagnosis ability for each fault. However, the overall average error is reduced as shown in Figure 14.

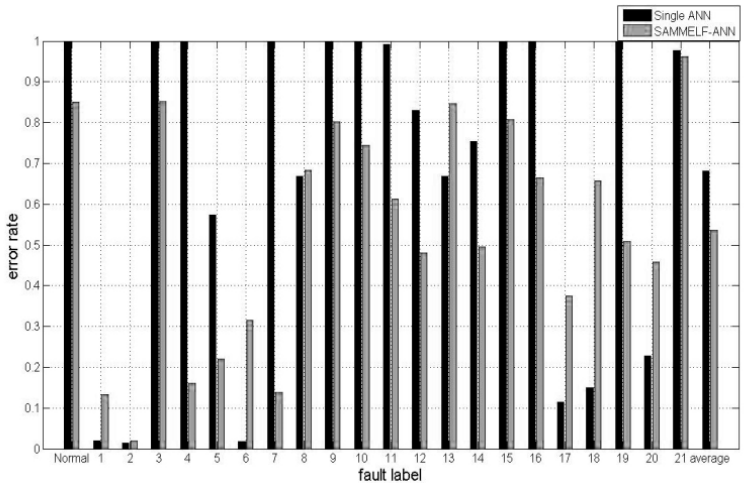

Figure 14. Classification performances of the single and boosted networks obtained by the SAMMELF algorithm on the test data under different fault conditions

5. Conclusion

In this work, three networks used for fault diagnosis of a chemical plant are compared to each other. Boosting as a general method employed to improve the classification performance of learning algorithms is utilized. The main idea in the boosting method is to enforce the basic classifier to work on the hard examples in a sequential rounds and then to combine the obtained single classifiers in an ensemble system. AdaBoost algorithm as the most popular boosting method is employed and its advantages and limitations in practical applications of process fault diagnosis are discussed. AdaBoost guarantees that the error of the composite classifier on the training data decreases exponentially to a very small value. However, the convergence is achieved if the weak learner (MLP neural networks in our application) has always prediction error less than 0.5 in each boosting stage. Whereas, this condition may not always be obtainable especially in applications where a large number of faults is encountered, a substitute boosting method called SAMMELF is also utilized. The SAMMELF algorithm follows closely the philosophy of boosting and shares the same simple modular structure of AdaBoost. However, its convergence is achieved by imposing a relatively relaxed condition on each weak learner prediction error which normally requires error to be less than a random guessing.

The TEP simulator as a well-known challenging diagnosis benchmark is utilized in this work to assess three different fault classification methodologies. The performances of the single neural network and the ensemble model obtained by the AdaBoost and SAMMELF algorithms are compared against each other under different conditions. The results showed that using a boosting based ensemble method always led to an improved diagnosis performance. The employed boosting methods were able to exponentially reduce the training error without generating over-fitted classification models. In addition, the generalization error is improved in all cases. However, the AdaBoost algorithm failed when boosting the network dealing with the whole TEP diagnosis problem because this network always produced errors more than 0.5 on the training data set. Consequently, the conventional AdaBoost algorithm may not always be applied to complex diagnosis problems where numerous process faults are presented. On the other hand, the SAMMELF algorithm performed well in these difficult diagnosis situations and always improved the fault diagnosis performance in the sequential rounds of boosting. Hence, it presents a proficient boosting scheme which is more suitable for industrial process fault diagnosis.

References

[1] Himmelblau, D.M., *Fault detection and diagnosis in chemical and petrochemical processes*, Elsevier Scientific Publishing Company, Amsterdam, 1978.

[2] Chiang, L.H., Braatz, R.D. and Russell, E.L., *Fault detection and diagnosis in industrial systems*, Springer, New York, 2001.

[3] Venkatasubramanian, V., Rengaswamy, Yin, R.K. and Kavuri, S.N., "A review of process fault detection and diagnosis: Part I: Quantitative model-based methods," *Computers & chemical engineering*, 27 (3). 293-311. 2003.

[4] Isermann, R., "Process fault detection based on modeling and estimation methods-a survey," *Automatica*, 20 (4). 387-404. 1984.

[5] Dash, S. and Venkatasubramanian, V., "Challenges in the industrial applications of fault diagnostic systems," *Computers & chemical engineering*, 24 (2). 785-791. 2000.

[6] Kresta, J.V., Macgregor, J.F. and Marlin, T.E., "Multivariate statistical monitoring of process operating performance," *The Canadian journal of chemical engineering*, 69 (1). 35-47. 1991.

[7] He, Q.P. and Wang, J., "Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes," *Semiconductor manufacturing, IEEE transactions on*, 20 (4). 345-354. 2007.

[8] Chiang, L.H., Kotanchek, M.E. and Kordon, A.K., "Fault diagnosis based on Fisher discriminant analysis and support vector machines," *Computers & chemical engineering*, 28 (8). 1389-1401. 2004.

[9] Verron, S., Tiplica, T. and Kobi, A., "Bayesian networks and mutual information for fault diagnosis of industrial systems," in *Workshop on Advanced Control and Diagnosis (ACD'06)*, 2006.

[10] Verron, S., Tiplica, T. and Kobi, A., "Fault diagnosis of industrial systems by conditional Gaussian network including a distance rejection criterion," *Engineering applications of artificial intelligence*, 23 (7). 1229-1235. 2010.

[11] Chen, K.Y., Chen, L.S., Chen, M.C. and Lee, C.L., "Using SVM based method for equipment fault detection in a thermal power plant," *Computers in industry*, 62 (1). 42-50. 2011.

[12] Yélamos, I., Escudero, G., Graells, M. and Puigjaner, L., "Performance assessment of a novel fault diagnosis system based on support vector machines," *Computers & chemical engineering*, 33 (1). 244-255. 2009.

[13] Kulkarni, A., Jayaraman, V.K. and Kulkarni, B.D., "Knowledge incorporated support vector machines to detect faults in Tennessee Eastman process," *Computers & chemical engineering*, 29 (10). 2128-2133. 2005.

[14] Xu, J., Zhao, J., Ma, B. and Hu, S., "Fault diagnosis of complex industrial process using KICA and sparse SVM," *Mathematical problems in engineering*, 2013. 1-6. 2013.

[15] Hussain, M. A., Hassan, C.R.C., Loh, K.S. and Mah, K.W., "Application of artificial intelligence technique in process fault diagnosis," *Journal of engineering science and technology*, 2 (3). 260-270. 2007.

[16] Watanabe, K., Matsuura, I., Abe, M., Kubota, M. and Himmelblau, D.M., "Incipient fault diagnosis of chemical processes via artificial neural networks," *AIChE journal*, 35 (11). 1803-1812. 1989.

[17] Koivo, H.N., "Artificial neural networks in fault diagnosis and control," *Control engineering practice*, 2 (1). 89-101. 1994.

[18] Himmelblau, D.M., "Applications of artificial neural networks in chemical engineering," *Korean journal of chemical engineering*, 17 (4). 373-392. 2000.

[19] Sharma, R., Singh, K., Singhal, D. and Ghosh, R., "Neural network applications for detecting process faults in packed towers," *Chemical engineering and processing: process intensification*, 43 (7). 841-847. 2004.

[20] Eslamloueyan, R., "Designing a hierarchical neural network based on fuzzy clustering for fault diagnosis of the Tennessee–Eastman process," *Applied soft computing*, 11 (1). 1407-1415. 2011.

[21] Behbahani, M.R., Jazayeri-Rad, H. and Hajmirzaee, S., "Fault detection and diagnosis in a sour gas absorption column using neural networks," *Chemical engineering & technology*, 32 (5). 840-845. 2009.

[22] Hansen, L.K. and Salamon, P., "Neural network ensembles," *Pattern analysis and machine intelligence, IEEE transactions on*, 12 (10). 993-1001. 1990.

[23] Polikar, R., "Ensemble based systems in decision making," *Circuits and systems magazine, IEEE*, 6 (3). 21-45. 2006.

[24] Baxt, W.G., "Improving the accuracy of an artificial neural network using multiple differently trained networks," *Neural computation*, 4 (5). 772-780. 1992.

[25] Ali, K.M. and Pazzani, M.J., "Error reduction through learning multiple descriptions," *Machine learning*, 24 (3). 173-202. 1996.

[26] Valdovinos, R.M. and Sanchez, J. S., "Ensembles of multilayer perceptron and modular neural networks for fast and accurate learning," in *Artificial Intelligence, MICAI'06. Fifth Mexican International Conference on*, 2006, 229-236.

[27] Zhang, J., "Improved on-line process fault diagnosis through information fusion in multiple neural networks," *Computers & chemical engineering*, 30 (3). 558-571. 2006.

[28] Bauer, E. and Kohavi, R., "An empirical comparison of voting classification algorithms: bagging, boosting, and variants," *Machine learning*, 36 (1). 105-139. 1999.

[29] Zio, E., Baraldi, P. and Gola, G., "Feature-based classifier ensembles for diagnosing multiple faults in rotating machinery," *Applied soft computing*, 8 (4). 1365-1380. 2008.

[30] Xu, L., Krzyzak, A. and Suen, C.Y., "Methods of combining multiple classifiers and their applications to handwriting recognition," *Systems, man and cybernetics, IEEE transactions on*, 22 (3). 418-435. 1992.

[31] Kuncheva, L.I., Bezdek, J.C. and Duin, R.P.W., "Decision templates for multiple classifier fusion: an experimental comparison," *Pattern recognition*, 34 (2), 299-314, 2001.

[32] Ruta, D. and Gabrys, B., "An overview of classifier fusion methods," *Computing and Information systems*, 7 (1). 1-10. 2000.

[33] Freund, Y. and Schapire, R.E., "Experiments with a new boosting algorithm," in *ICML*, 1996, 148-156.

[34] Freund, Y. and Schapire, R.E., "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory*, 1995, 23-37.

[35] Schwenk, H., and Bengio, Y., "Boosting neural networks," *Neural Computation*, 12 (8). 1869-1887. 2000.

[36] Schapire, R.E., Freund, Y., Bartlett, P. and Lee, W.S., "Boosting the margin: A new explanation for the effectiveness of voting methods," *The annals of statistics*, 26 (5). 1651-1686. 1998.

[37] Samanta, B., Bandopadhyay, S., Ganguli, R. and Dutta, S., "A comparative study of the performance of single neural network vs. adaboost algorithm based combination of multiple neural networks for mineral resource estimation," *Journal of South African institute of mining and metallurgy*, 105 (4). 237-246. 2005.

[38] Schwenk, H. and Bengio, Y., "Adaboosting neural networks: Application to on-line character recognition," in *Artificial Neural Networks-ICANN'97*, Springer, 1997, 967-972.

[39] Canty, M.J., "Boosting a fast neural network for supervised land cover classification," *Computers & geosciences*, 35 (6). 1280-1295. 2009.

[40] Zhu, J., Zou, H., Rosset, S. and Hastie, T., "Multi-class adaboost," *Statistics and Its Interface*, 2. 349-360. 2009.

[41] Downs, J.J. and Vogel, E.F., "A plant-wide industrial process control problem," *Computers & chemical engineering*, 17 (3). 245-255. 1993.

[42] Lyman, P.R. and Georgakis, C., "Plant-wide control of the Tennessee Eastman problem," *Computers & chemical engineering*, 19 (3). 321-331. 1995.

[43] Møller, M.F., "A scaled conjugate gradient algorithm for fast supervised learning," *Neural networks*, 6 (4). 525-533. 1993.

[44] Mahadevan, S. and Shah, S.L., "Fault detection and diagnosis in process data using one-class support vector machines," *Journal of process control*, 19 (10). 1627-1639. 2009.