Cold Start Approach for Data-Driven Fault Detection

Mihajlo Grbovic, Weichang Li, Niranjan A Subrahmanya, Adam K Usadi, Slobodan Vucetic

Abstract— **A typical assumption in supervised fault detection is that abundant historical data are available prior to model learning, where all types of faults have already been observed at least once. This assumption is likely to be violated in practical settings as new fault types can emerge over time. In this paper we study this often overlooked cold start learning problem in data-driven fault detection, where in the beginning only normal operation data are available and faulty operation data become available as the faults occur. We explored how to leverage strengths of unsupervised and supervised approaches to build a model capable of detecting faults even if none are still observed, and of improving over time, as new fault types are observed. The proposed framework was evaluated on the benchmark Tennessee Eastman Process data. The proposed fusion model performed better on both unseen and seen faults than the stand-alone unsupervised and supervised models.**

***Index Terms*— cold start learning, fault detection, process monitoring, semi-supervised learning.**

I. Introduction

Data-Driven Fault Detection has been extensively studied during the past few decades [1-20]. Timely discovery of faulty events in complex systems is critical to ensure safety and support effective operation. The objective of fault detection is to achieve high fault detection accuracy with low detection delay. Early detection provides an invaluable warning on emerging problems that can be, relatively speaking, easily managed to avoid catastrophic consequences. High accuracy ensures that the human operator is rarely interrupted by false alarms and guarantees successful detection and tracking of fault events.

There are two major approaches for data driven fault detection. In the unsupervised approach, normal operation is modeled and faults are detected as deviations from the normal behavior. Some of the most popular unsupervised fault detection methods are Principal Component Analysis (PCA) [6, 7], Independent Component Analysis (ICA) [9] and Partial Least Squares [6]. In the supervised approach, a classifier is trained on annotated historical data containing both normal and faulty conditions, and it is used to predict faults. Representatives of this approach are Support Vector Machines (SVM) [11] and Neural Networks [1]. Due to annotation expenses, historical data may consist of only a small amount of labeled data and a large amount of unlabeled data. In such cases, a semi-supervised learning approach can be used [21], which is class of machine learning techniques that makes use of both labeled and unlabeled data for training.

The unsupervised models are practical because they can be constructed using small amounts of normal condition data, whereas supervised models can achieve increased sensitivity to faults and provide better accuracy, but assume that annotated data containing all fault types are available for training. This assumption is likely to be violated in real-world applications.

In this paper we study an important problem of *cold start* learning, where only limited amounts of normal operation data are initially available for fault detection model training. As knowledge about different fault types becomes available, in form of new partially or fully annotated data, the goal is to incrementally update the model to improve detection accuracy.

To illustrate, let us assume that a manufacturer of certain type of gas turbines tests each turbine under close human supervision to ensure safe operation before the turbines are sold and used. In this process, the manufacturer can collect a large amount of normal operation data, sufficient to build a single unsupervised fault detection model. After the turbines are sold and put to use, the manufacturer continues monitoring each turbine to collect data of unforeseen faulty operation that can be useful for improving the current fault detection model through supervised learning. Training and updating of a supervised fault detection model using faulty operation data from any monitored turbine can help in timely detection and avoidance of the same fault types on the remaining turbines.

Leveraging strengths of unsupervised and (semi-)supervised fault detection models in the cold start scenario has not been extensively studied in the literature. There are several recent publications in which the supervised and unsupervised models are combined to improve fault detection accuracy. However, in most cases [13-17], the unsupervised model is used only as a preprocessing tool to extract better features for the supervised model. In some publications, the supervised model is used to

Mihajlo Grbovic and Slobodan Vucetic are with the Department of Computer and Information Sciences, Temple University, 1805 N Broad Street 304 Wachman Hall Philadelphia, PA, USA 19122 (phone: 215-204-5535; fax: 215-204-5082; e-mail: mihajlo.grbovic@temple.edu).

Weichang Li, Niranjan A Subrahmanya, Adam K Usadi are with the ExxonMobil Research and Engineering Company, Corporate Strategic Research, Annandale, NJ, USA 08801.

provide fault localization once the fault is detected by the unsupervised model [19, 14]. In addition, there exist decentralized approaches [22] where, for various reasons, groups of features, e.g. sensors, are monitored separately, some using unsupervised model and some using supervised model.

In this study we propose an approach that integrates decisions from the initial unsupervised model and an incrementally updated supervised model, which leads to overall improvement in fault detection accuracy. We propose several strategies for combining the detectors. In addition, to deal with partially annotated data, we propose a semi-supervised framework that facilitates learning from partially annotated data via minimization of the disagreement between detectors in predicting the fault class

II. Preliminaries

A. Problem Setup

We consider M identical plants or pieces of equipment, each monitored separately by a network of K sensors providing measurements synchronously at regular time intervals. For m-th plant at time t, the state of its k-th sensor is represented by a row vector of variables $\mathbf{x}_{tk}{}^{m}$. There are many ways to construct this vector. For example, one can use only raw measurements at time t, the set of raw measurements at the most recent t_{lag} time steps, or derive variables from the current and recent measurements. Combining all K sensors, we have a row vector $\mathbf{x}_t{}^m = [\mathbf{x}_{t1}{}^m, \mathbf{x}_{t2}{}^m, \ldots, \mathbf{x}_{tK}{}^m]$. We denote with d the length of the resulting vector. The process condition at m-th plant at time t is denoted with class label $c_t{}^m \in \{-1, +1\}$, where -1 represents normal condition, and $+1$ represents a fault. Combining all monitored plants, we have a collection $D_{tr} = \{(\mathbf{x}_i, c_i), i = 1 \ldots N\}$.

Given the data D_{tr}, the objective is to train a classification function $f\colon \mathbf{x}_i \rightarrow \hat{c}_i$, where $\hat{c}$ is the hard prediction output $\hat{c}_i \in \{-1, +1\}$. In some cases, function $g: \boldsymbol{x}_i \rightarrow \hat{y}_i$ that outputs a value $\hat{y}_i \in \mathrm{R}$ is used for soft prediction. In that case, $\hat{c}_i$ can be obtained by thresholding, $\hat{c}_i = sign(\hat{y}_i - \theta)$, where θ is the specified threshold.

In the cold start scenario, we assume that N_{int} normal condition observations, $D_{\boldsymbol{int}} = \{(\mathbf{x}_i, -1),\ i = 1 \ldots N_{int}\}$, are available for development of an unsupervised detector.

By continuous monitoring of all processes, the faulty operation data become available as the faults occur. Whenever a new fault type is observed at any plant, data directly before and after the fault occurrence are collected. Figure 1 shows an example of such a data batch B_j from one of the plants, which contains L_j observations. Assuming the fault occurred at time $t_j{}^0$, the first $t_j{}^0-1$ examples are considered as normal operation data, while the remaining $L_j - t_j{}^0+1$ examples are considered as faulty operation data. Depending on the particular application, all or parts of data B_j are labeled with $+1$ or -1. Specifically, B_j can be divided into the labeled part $\{(\mathbf{x}_i, c_i), i = 1 \ldots l\}$ and the unlabeled part $\{\mathbf{x}_i, i = l+1 \ldots l+u\}$, $l + u = L_j$.

At any time t, the objective is to train a classifier using the available data, $D_{tr} = D_{\boldsymbol{int}} \cup B_{\mathrm{j}} \cup \ldots \cup B_J$.

B. Performance Measures

To evaluate performance of fault detection models, we will use a set of labeled observations, disjoint from the training set, consisting of both normal and faulty conditions.

The *true positive rate, TPR,* is defined as

$$TPR = \frac{n^+}{N^+} \cdot 100, \tag{1}$$

where n^+ is the number of correctly classified faulty operation examples and N^+ is the total number of faulty operation examples in test data.

The *false positive rate*, FPR, is given as

$$FPR = \frac{n^-}{N^-} \cdot 100, \tag{2}$$

where n^- is the number of misclassified normal operation data examples and N^- is the total number of normal operation data examples in test data

Let us assume test data contain J time series in which normal and faulty conditions interchange, as in Figure 1. The flat line is the true label and the dotted line is an example fault detector prediction. We define the *detection delay DD_j* for the j-th time series from test data as the delay between $t_j{}^0$, the introduction time of the fault, and $t_j{}^1$, the model detection time,

$$DD_j = t_j^1 - t_j^0. \tag{3}$$

The average detection delay for all time series in test data is

$$DD = \sum_{j=1}^{J} \frac{DD_j}{J}. \tag{4}$$

III. Driven Fault Detection

In this section, we describe the unsupervised, supervised and semi-supervised methods that we used as components of the fusion model proposed later in this paper.

A. Unsupervised Fault Detection

In the unsupervised approach, faults are detected as observations that deviate from the modeled normal behavior. Among the most popular unsupervised fault detection methods are PCA and ICA-based approaches. They are threshold-based approaches that input the observation vector and output a single number representing deviation from the normal behavior. The

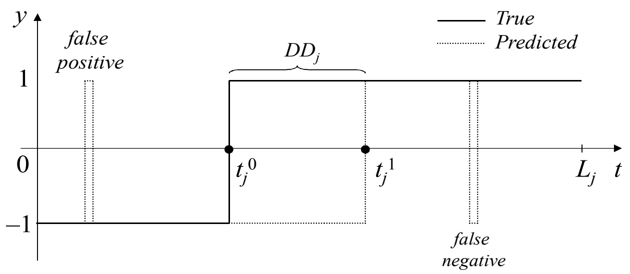


Fig. 1. Batch B_j containing a faulty event: true vs. predicted

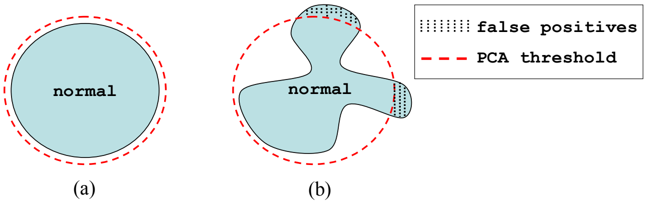


Fig. 2. Unsupervised model: (a) simple (b) complicated distribution

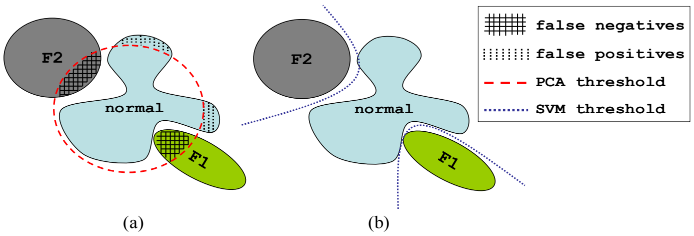


Fig. 3. Complicated distribution a) unsupervised model b) supervised model

threshold is typically set such that FPR is small (in the range between 1% and 5%).

The PCA-based approach [6] has been extensively studied and employed for fault detection. PCA projects the observation to a lower dimensional subspace, which describes most of the variance in the normal data. Given training data consisting of N centered normal condition observations, let us define $N\times d$ matrix $\mathbf{X}=[\mathbf{x}_1^{\mathrm{T}}, \mathbf{x}_2^{\mathrm{T}}, \ldots, \mathbf{x}_N^{\mathrm{T}}]^{\mathrm{T}}$, where $\mathbf{x}_i$ is a d-dimensional row vector of the i-th observation. Then, the principal components are found using eigenvalue decomposition of the covariance matrix $\mathbf{S}$,

$$\mathbf{S}=\frac{1}{N-1}\mathbf{X}^T\mathbf{X}=\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \tag{5}$$

where the columns of $\mathbf{V}=[\mathbf{v}_1, \mathbf{v}_2,\ldots,\mathbf{v}_d]$ are the eigenvectors of $\mathbf{S}$ and the diagonal elements of $\mathbf{\Lambda}$ are its eigenvalues. Two types of statistics are typically used in PCA–based process monitoring; namely, Hotelling's T^2 statistic and the squared prediction error (SPE) statistic [7].

The SPE statistics, which are used in our experiments, is a measure of variation in the residual subspace, defined as

$$SPE(i)=\mathbf{x}_i^{\,T}\mathbf{V}_{d-a}\mathbf{V}_{d-a}^T\mathbf{x}_i, \tag{6}$$

where $\mathbf{V}_{d\text{-}a}$ is a set of the $d-a$ smallest eigenvectors of $\mathbf{S}$. Reduction order a ($a<d$) is directly related to the percentage of variance retained in the normal condition data.

The fault is reported if the value of SPE statistic exceeds the threshold level θ_u which is determined by the desired FPR.

The unsupervised models are practical and quickly operational, as they can construct a fault detector using small amounts of normal operation data. However, there are known performance issues, as these models inherently assume Gaussian distribution of normal operation data. Figure 2.a shows an idealized case where, after mapping to SPE statistic space, normal operation data are within a circle, due to Gaussian distribution in the input space. In this case, data can be adequately described using PCA SPE statistic threshold. However, mapping of more complex normal operation distributions to SPE statistic space can have results such as in Figure 2.b. Using PCA SPE statistic threshold in this case can lead to a large fraction of false positives and false negatives.

B. Supervised Fault Detection

Supervised methods, such as Support Vector Machine (SVM) need both normal and faulty conditions data to train a fault detector. The advantage of the supervised methods is that they explicitly utilize historical information of the faulty conditions, which increases their sensitivity to faults. Given faulty data, better performance over the unsupervised models can be achieved because more involved decision rules that maximize TPR are possible, as illustrated in Figure 3. Normal conditions and two fault types (F1 and F2) are shown. As can be observed, the unsupervised model (Figure 3.a) has a large number of false negatives and false positives, whereas the supervised model (Figure 3.b) achieves much higher accuracy.

SVM. Given a data set D_{tr} = {($\mathbf{x}_i$, c_i), $i=1\ldots N$}, SVM classifier [23, 24] attempts to find the maximum-margin hyperplane that divides the examples of the opposite classes. SVM seeks the best classifier of type $g(\mathbf{x})=w^{\mathrm{T}}\Phi(\mathbf{x})+\beta$, where Φ: $\Re^d\rightarrow H$ is a mapping from the original d-dimensional attribute space to a potentially high-dimensional space H. Maximizing the margin is equivalent to minimizing $\|w\|$, and the problem can be formulated as

$$\begin{aligned}&\min_{w,b}.\frac{1}{2}\|w\|^2+C\sum_{i=1}^{N}\xi_i\\&s.t.\;y_i(w\cdot\Phi(\mathbf{x}_i)+b)\geq 1-\xi_i,\,\xi_i\geq 0,\forall i\,,\end{aligned} \tag{7}$$

where ξ_i are the slack variables, introduced to account for noise and non-separable data, and $C>0$ is a penalty parameter that trades-off model complexity and accuracy on training data. Problem (7) can be converted to dual form,

$$\begin{aligned}&\min_{0\leq\alpha_i\leq S}:\sum_i\alpha_i-\frac{1}{2}\sum_i\sum_j\alpha_i\alpha_j y_i y_j\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)\\&s.t.\;\;\sum_i\alpha_i y_i=0,\,0\leq\alpha_i\leq C\;\forall i\qquad,\end{aligned} \tag{8}$$

where α_i are the Lagrange multipliers associated with the constraints of the primal problem.

The resulting SVM classifier can be conveniently represented using the dual problem solution as

$$g(\mathbf{x})=\sum_{i=1}^{N}y_i\alpha_i\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x})=\sum_{i=1}^{N}y_i\alpha_i\mathbf{K}(\mathbf{x}_i,\mathbf{x}), \tag{9}$$

where $\mathbf{K}(\cdot,\cdot)$ is the kernel function induced by Φ and α_i are the Lagrange multipliers associated with the training examples.

Given soft SVM outputs $\hat{y}_i=g(\mathbf{x}_i)$, the final fault detection predictions are made as $\hat{c}_i=f(\mathbf{x}_i)=sign(\hat{y}_i-\theta_s)$, where threshold θ_s is set to desired FPR.

C. Semi-supervised Fault Detection

Semi-supervised methods are typically used when majority of the training data D_t is unlabeled. Such methods aim at simultaneously achieving high accuracy on the labeled portion of data and ensuring some geometric dependence on unlabeled examples. A number of graph-based semi-supervised learning algorithms [25, 26, 27] can be placed under the framework of

f_u	+1	+1	+1	+1	+1	+1
f_{ss}	–1	–1	–1	–1	–1	–1

a)

f_u	–1	–1	–1	–1	–1	–1
f_{ss}	+1	+1	+1	+1	+1	+1

b)

Fig. 4. Unsupervised and semi-supervised model predictions

minimizing a loss function that involves both labeled and unlabeled examples,

$$\min_{g_{ss}} \sum_{i=1}^{l} Loss(g_{ss}(\mathbf{x}_i), c_i) + C_1\|g_{ss}\|_K^2 + C_2\mathbf{g}_{ss}^{\mathbf{T}}\mathbf{L}\mathbf{g}_{ss}, \qquad (10)$$

where $g_{ss}: \boldsymbol{x}_i \rightarrow \hat{y}_i$, $\mathbf{g}_{ss} \in \mathrm{R}^{n\times 1}$ are semi-supervised model soft predictions for all samples, with the labeled part $\mathbf{g}^{l}_{ss} \in \mathrm{R}^{l\times 1}$ and the unlabeled part $\mathbf{g}^{u}_{ss} \in \mathrm{R}^{u\times 1}$, $\mathbf{K} \in \mathrm{R}^{n\times n}$ is a kernel matrix describing the similarity between examples in the feature space, $\mathbf{D} = diag(\mathbf{K}\,\mathbf{1}_n)$ is the degree matrix, and $\mathbf{L}$ is the regularization Laplacian matrix, $\mathbf{L} = \mathbf{D} - \mathbf{K}$.

The first term enforces that predictions $\mathbf{g}_{ss}$ should be consistent with the known class labels. The second term is a regularization term, where $\|g_{ss}\|_K$ is the Reproducing Kernel Hilbert Space (RKHS) norm of prediction function g_{ss}, and C_1 is the associated regularization parameter. The third term enforces a geometric constraint that the predictions should be sufficiently consistent with the structure of the data in the space defined by kernel $\mathbf{K}$. It is known as the Laplacian regularization term [28] and can also be written as

$$\mathbf{g}_{ss}^{\mathbf{T}}L\mathbf{g}_{ss} = \sum_{i,j=1}^{l+u}\left(\frac{g_{ss}(\mathbf{x}_i)}{\sqrt{\mathbf{D}(i,i)}} - \frac{g_{ss}(\mathbf{x}_j)}{\sqrt{\mathbf{D}(j,j)}}\right)\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j). \qquad (11)$$

The minimizer of the problem (10) admits the expansion over all the labeled and unlabeled examples in the form of

$$g_{ss}(\mathbf{x}) = \sum_{i=1}^{l+u}\alpha_i\mathbf{K}(\mathbf{x}_i, \mathbf{x}), \qquad (12)$$

where α_i are the kernel expansion coefficients.

Prototype Vector Machine (PVM). Recent advances [26] allow for fast optimization of (10), by low-rank approximation of $\mathbf{K}$, and fast decision making, by forcing the decision function to span over a small set of basis vectors.

The resulting PVM algorithm uses a squared loss for computationally efficient learning and ease of implementation.

Using the representer theorem [28], $\mathbf{g}_{ss} = \mathbf{K}\alpha$, and assuming squared loss, $Loss(\cdot) = \|\cdot\|^2$, objective (10) can be rewritten as

$$\min_{\alpha\in R^{n\times 1}}\|\mathbf{Y}_l - \mathbf{K}_l\alpha\|^2 + C_1\alpha^T\mathbf{K}\alpha + C_2\alpha^T\mathbf{K}^T\mathbf{L}\mathbf{K}\alpha, \qquad (13)$$

where $\mathbf{Y}_l \in \mathrm{R}^{l\times 1}$ are the class label assignments for the labeled examples and $\mathbf{K}_l \in \mathrm{R}^{l\times n}$ are the rows in the kernel matrix corresponding to the labeled samples.

PVM for Fault Detection. In our specific setup, additional information is available in form of unsupervised model f_u predictions. This allows us to impose an extra constraint to better utilize the unlabeled examples. Our main assumption is that unlabeled samples in f_u view and f_{ss} view should agree in labels, $f_u(\mathbf{x}_i) = f_{ss}(\mathbf{x}_i)$, where $f_{ss}(\mathbf{x}) = sign(g_{ss}(\mathbf{x}))$.

However, as $f_{ss}(\mathbf{x})$ is expected to outperform f_u in accuracy of +1 predictions as more fault types are observed, the requirement of equal labels might be harmful for f_{ss}. To illustrate, let us consider the example in Figure 4. Figure 4.*a* can be interpreted as f_u detecting a fault type that f_{ss} has not previously observed. Figure 4.*b* depicts a scenario in which f_{ss} is simply outperforming f_u on a familiar fault type. For this reason, utilizing f_u is advantageous only on fault types not yet observed by f_{ss}. Therefore, we add the following regularization term into the optimization problem (13), weighted by C_3,

$$\min_{g_{ss}}\sum_{i=1}^{l+u}\left(\frac{f_u(\mathbf{x}_i)+1}{2}(g_{ss}(\mathbf{x}_i) - f_u(\mathbf{x}_i))^2\right), \qquad (14)$$

where $f_u(\mathbf{x}_i) \in \{-1,+1\}$ predictions are known and $g_{ss}(\mathbf{x}_i) \in \mathrm{R}$. The regularization is active only when $f_u(\mathbf{x}) = +1$. To preserve efficient optimization of PVM, a squared norm was used.

It should be noted that this framework can be used with any unsupervised model and any semi-supervised model capable of minimizing the resulting loss function,

$$\begin{aligned}\min_{\alpha\in R^{n\times 1}} \alpha^T\mathbf{K}_l^T\mathbf{K}_l\alpha - 2\mathbf{Y}_l^T\mathbf{K}_l\alpha + C_1\alpha^T\mathbf{K}\alpha + C_2\alpha^T\mathbf{K}^T\mathbf{L}\mathbf{K}\alpha \\ + C_3(\alpha^T\mathbf{K}^T\mathbf{A}^T\mathbf{A}\mathbf{K}\alpha - 2\mathbf{f}_u^T\mathbf{A}^T\mathbf{A}\mathbf{K}\alpha),\end{aligned} \qquad (15)$$

where $\mathbf{A}$ is the $n\times n$ diagonal matrix with $(f_u(\mathbf{x}_i) + 1)/2$ on the diagonal and $\mathbf{f}_u$ is a binary vector of unsupervised model predictions $f_u(\mathbf{x}_i) \in \{-1,+1\}$ for both labeled and unlabeled data.

IV. Fusion Model for Cold Start Learning

A realistic scenario in process monitoring is that at the time of fault detector training, we did not observe all types of faults. In the cold start case, we might only have access to the normal operation data. Using the initial data, unsupervised model can be trained. Upon occurrence of the first fault type it becomes possible to learn a supervised or a semi-supervised model, depending on the application at hand. As the process is experiencing new fault types, it should be possible to incrementally improve the (semi-)supervised model.

While the updated model could have reduced FPR on seen fault types, it could actually increase FPR on the unseen faults. Figure 5 illustrates this potential problem. Supervised model decision rule obtained from training data that includes faults F1 and F2 is shown. A new, previously unobserved fault type, fault F3, remains undetectable by this model.

As the example from Figure 6 suggests, it might be better to consult both unsupervised and supervised models. The unsupervised model can detect the three faults with higher overall accuracy, although the supervised model has higher TPR on F1 and F2. We can achieve higher overall TPR on the three faults by combining decisions of the supervised and unsupervised models using the “or” rule (if one of them detects the fault the fault is predicted).

Based on this intuitive observation, in the following we describe different strategies for combining predictions of unsupervised and supervised models into a single prediction. The unsupervised model can be combined with a

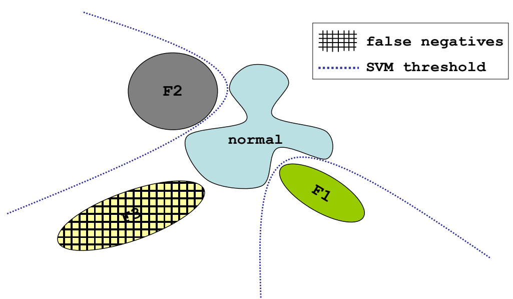


Fig. 5. Supervised model trained on F1, F2. Newly observed F3 is undetectable

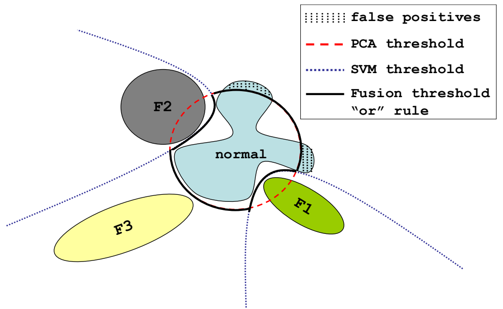


Fig. 6. Unsupervised and supervised models combined using "or" rule

semi-supervised model in the same manner. We also describe strategies for updating the fusion model in cases of completely and partially labeled data.

A. Combining Unsupervised and Supervised Models

1. "Or" rule. The fault is reported if either unsupervised model f_u or supervised model f_s detects the fault, $\hat{c}_i = sign(f_u(\mathbf{x}_i) + f_s(\mathbf{x}_i) + 1)$, where $f_u(\mathbf{x}_i), f_s(\mathbf{x}_i) \in \{-1, +1\}$. Due to the voting mechanism (even one +1 vote wins), the "or" rule could potentially increase the overall FPR as compared to FPR of the individual models. Essentially, the fusion model will inherit all of unsupervised and supervised models false alarms. However, the "or" rule will assure the effective detection of faults previously unseen by the supervised model, and increase the overall TPR, due to the supervised model superior TPR on the fraction of faults it has observed.

2. Performance-based Weighted Voting [29]. This approach assigns different weights to supervised and unsupervised classifiers based on their estimated true positive and false positive rates, TPR$_s$ and FPR$_s$ for the supervised model and TPR$_u$ and FPR$_u$ for the unsupervised model. Those rates are estimated using the available training data. Then, it calculates the prediction-dependent model weights as

$$w_s(\mathbf{x}_i) = \frac{(TPR_s - FPR_s)}{(1 - f_s(\mathbf{x}_i)) + f_s(\mathbf{x}_i) \cdot (FPR_s + TPR_s)}, \qquad (16)$$

$$w_u(\mathbf{x}_i) = \frac{(TPR_u - FPR_u)}{(1 - f_u(\mathbf{x}_i)) + f_u(\mathbf{x}_i) \cdot (FPR_u + TPR_u)}.$$

Finally, it calculates the final prediction as $\hat{c}_i = sign(w_u(\mathbf{x}_i) \cdot f_u(\mathbf{x}_i) + w_s(\mathbf{x}_i) \cdot f_s(\mathbf{x}_i))$. When both models agree, the final prediction is obtained by consensus. When the models disagree, the model with the larger weight decides.

3. Distance-based Weighted Voting. Considering soft predictions from supervised and unsupervised models, $\hat{y}_u$, $\hat{y}_s \in$ R, and their distances from the corresponding thresholds, $\hat{y}_u - \theta_u$ and $\hat{y}_s - \theta_s$, a heuristic weighted voting schema can be derived as $\hat{c}_i = sign((g_u(\mathbf{x}_i) - \theta_u) + (g_s(\mathbf{x}_i) - \theta_s))$. Interpreting distance from the threshold as an estimate of uncertainty, this voting schema always trusts the model that is more certain in its prediction.

4. Maximum Entropy Voting. Discriminative probabilistic Maximum Entropy (MaxEnt) model [30] can be used for combining local model predictions into a global prediction [31] by introducing pre-defined decision making rules.

In our setting, we propose to use rules that involve both models and the predicted class, i.e. $r(f_u, f_s, c)$. For example, the following rule describes an outcome in which the unsupervised model predicts faulty operation, $f_u(\mathbf{x}_i) = +1$, the supervised model predicts normal operation $f_s(\mathbf{x}_i) = -1$ and the ground truth is $c_i = +1$.

$$r_1(f_u(\mathbf{x}_i), f_s(\mathbf{x}_i), c_i) = \begin{cases} 1, & f_u(\mathbf{x}_i) = +1 \wedge f_s(\mathbf{x}_i) = -1 \wedge c_i = +1 \\ 0, & otherwise \end{cases}.$$

If these conditions are met, this rule is included in prediction making, $r_1 = 1$, otherwise it is not considered, $r_1 = 0$. A total of eight rules can be created in this manner (for each possible outcome). Each rule $r_j(f_u, f_s, c)$ is associated with a certain weight ω_j that characterizes its influence in prediction making, i.e. the rules with large ω are more influential in the final prediction. Using rules and their weights, MaxEnt model calculates conditional probability

$$P(c_i \mid f_u, f_s) = \frac{\exp(\sum_{j=1}^{J} \omega_j r_j(f_u, f_s, c_i))}{\sum_{z \in \{-1,1\}} \exp(\sum_{j=1}^{J} \omega_j r_j(f_u, f_s, z))}, \qquad (17)$$

where $r_j(f_u, f_s, c)$ is the j-th rule and ω_j is the weight of the j-th rule. Given local predictions $f_u(\mathbf{x}_i)$ and $f_s(\mathbf{x}_i)$, global prediction $\hat{c}$ is made as the one that maximizes the conditional probability

$$\hat{c}_i = \arg\max_c P(c \mid f_u(\mathbf{x}_i), f_s(\mathbf{x}_i)). \qquad (18)$$

Weights ω_j are learned by maximizing likelihood,

$$L_{ME} = \sum_{i=1}^{N} \log P(c_i \mid f_u(\mathbf{x}_i), f_s(\mathbf{x}_i)). \qquad (19)$$

Since L_{ME} is a concave function, there exists a global optimum solution that can be found using standard convex optimization algorithms. In this paper, we use the gradient ascent iterative procedure that updates current estimates of ω_j as

$$\omega_j^{new} = \omega_j^{old} + \eta \frac{\partial L_{IK}}{\partial \omega_j}, \qquad (20)$$

where ω_j are initially set to $\omega_j = 1/J$, $j = 1 \ldots J$, $J = 8$.

B. Incremental update of the Fusion Model

Following the setup in II.A, the procedure starts by using the initial normal-type data D_{int}, when the unsupervised model is developed. Upon arrival of the first fault data batch B_1, a supervised model is trained, which is followed by selection of an appropriate fusion rule. The process is repeated upon receipt of any subsequent data batch when the supervised model is

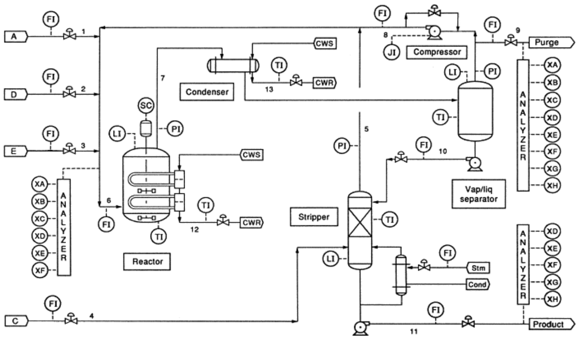


Fig. 7. Tennessee Eastman Process scheme

retrained and the fusion rule updated. Our approach allows for both supervised scenario where new data batches are completely annotated, and semi-supervised scenario, where the new batches are only partially annotated.

V. Empirical Studies

The proposed cold start fault detection framework with different decision fusion strategies was evaluated on the benchmark Tennessee Eastman Process (TEP) data set [4].

Two scenarios were considered. In the first scenario incoming data batches are completely annotated. PCA (using SPE statistic) and SVM were used as representatives of the unsupervised and supervised models, respectively. The resulting fusion model is denoted as PCA-SVM.

In the second scenario, the batches are partially labeled. PCA was used as the unsupervised model. Learning from partially labeled data was guided by optimization of the proposed objective (15), via PVM algorithm.

A. Tennessee Eastman Process

The Tennessee Eastman Process data set is a well-known simulated industrial problem for process monitoring and control. Over years, it has become a benchmark data for a large number of fault detection approaches [3, 5, 6, 7, 9, 11, 12]. The model is described in detail in [4]. The TEP model is a chemical process with five major operation units: a reactor, a condenser, a compressor, a stripper, and a separator (Figure 7). The plant represents an open-loop unstable plant that produces two liquid products G and H from gaseous feeds A, C, D, E, and the inert component B.

The process has 53 variables, including 22 process measurements, 19 analyzer measurements, and 12 manipulated variables. Our evaluation results presented here focus on all the process measurements and 11 manipulated variables, where the remaining constant manipulated variable was ignored. We did not consider the analyzer measurements.

The modified closed-loop version [3] of the original Fortran TEP implementation [4] was used to simulate data. The source code uses a discrete control algorithm to stabilize the process. A total of 20 types of faults (Table 1) can be simulated ranging from faults that are easy to detect with no delay (e.g., fault 1 and 4), to faults that are detectable only after a certain delay (e.g., fault 17 and 18), to very subtle faults that are hard to detect (faults 3, 9, 15) even with the best supervised methods. These properties of the TEP faults have been observed by previous studies [5-9, 11, 12]. As a result, our evaluation did not consider faults 3, 9 and 15.

fault	Description	type
1	A/C feed ratio, B composition constant (stream 4)	Step
2	B composition, A/C feed ratio constant (stream 4)	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 (str. 4)	Step
8	A, B, C feed composition (stream 4)	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
16-20	Unknown	

Table 1. Tennessee Eastman Process faults

B. Experimental Setup

We considered M=10 identical TEP plants at different locations. The goal was to train a single fault detection model for monitoring of all plants, through fusion of unsupervised and supervised learning. It was assumed that limited amounts of normal operation data from each plant D_{int} were initially available for unsupervised model training. Next, the plants were monitored for potential faulty behavior. Whenever a new fault type was observed at any location, data directly before and after the fault occurrence were collected in form of data batch B_m. Different fault types were randomly generated at different plants at different times, and data batches containing different fault types were observed as they occurred.

To create input attributes $\mathbf{x}_i$, we used the dynamic approach from [7, 9]. At each plant, observations from all K sensors at time t were augmented with observations from previous t_{lag} moments and stacked into a variable vector $\mathbf{x}_t^m$ with $(t_{lag}+1)\cdot K$ variables. Larger values of t_{lag} lead to improved performance with respect to FPR and TPR but increase the Detection Delay. As low Detection Delay was preferred, we used $t_{lag} = 2$.

Training Data for Unsupervised Model. Initial Data consisted of $N = 2{,}000$ labeled normal process operation examples, $D_{init} = \{(\mathbf{x}_i, c_i), c_i = -1, i = 1 \ldots N\}$.

Training Data for Supervised Model. We generated 17 fault types in a random order at different locations, such that it resulted in $M = 17$ labeled data batches $B_j = \{(\mathbf{x}_i, c_i), i = 1 \ldots L\}$, one for each fault type. Each batch was a time-series that contained $L = 4{,}000$ examples where the first 2,000 examples represented normal operation and the remaining ones represented faulty operation. The batches were used to update the supervised model, and to update the performance-based voting weights and MaxEnt weights.

Training Data for Semi-supervised Model. In the partially labeled data scenario, only 2.5% of randomly selected samples from each data batch were labeled.

Test Data. The performance was evaluated on test data consisting of 10 time-series for each of the 17 fault types were

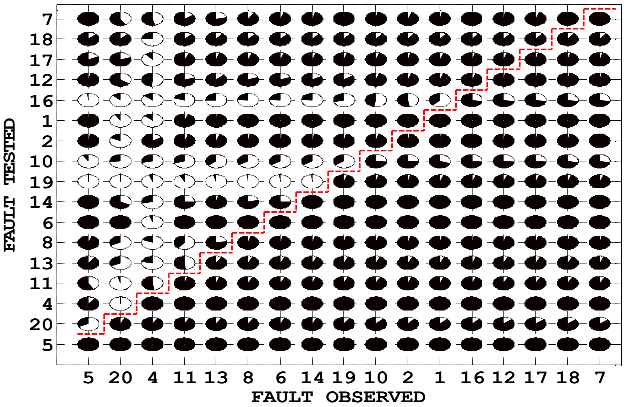


Fig. 8. Fault-by-fault TPR (y-axis) as new faults were observed (x-axis)

simulated. Each time-series contained $L=1{,}000$ examples, where first 500 examples represented normal operation and the remaining examples represented faulty operation.

Parameter Selection. Parameters for different algorithms were selected as follows. We assumed that no faulty data were available for parameter tuning.

a) **PCA**. SPE values for data D_{int} were calculated, i.e. the amount of variation in the residual subspace spanned by $d-a$ smallest principal components, where a is the number of principal components that explain 95% variance in data. Decision threshold θ_u was set such that FPR = 1%.

b) **SVM** used Gaussian RBF kernel $\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j)$ = $\exp(-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2)$, where the kernel width γ was set to inverse median of squared distances $\|\mathbf{x}_i-\mathbf{x}_j\|^2$ between data points in D_{int}, and slack parameter to $C=10$. LibSVM software package [32] was used. Decision threshold θ_s was set such that FPR = 1%.

c) **MaxEnt.** Gradient ascent learning rate was set to $\eta=0.001$ and the learning of rule weights ω was terminated when (20) stopped improving by more than 10^{-5}.

d) **PVM** used a Gaussian Kernel $\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2)$. The kernel parameter γ was set to inverse median of squared distances $\|\mathbf{x}_i-\mathbf{x}_j\|^2$ between data points in D_{int}. The number of PVM prototypes was set to 200. Regularization parameters in loss function (15) were set as follows, $C_1=10$, $C_2=0.1$ and $C_3=10$. Decision threshold θ_{ss} was set such that FPR = 1%.

The fusion model used individual models whose FPR was set in advance to 1%. Depending on the fusion rule the resulting FPR could possibly increase or decrease. For example, the “or” rule FPR can only be in the 1%-2% range, depending on how much the individual false alarms of SVM and PCA overlap, while the distance-weighted voting strategy could potentially achieve FPR that is less than 1%, since it uses soft predictions and prediction uncertainties.

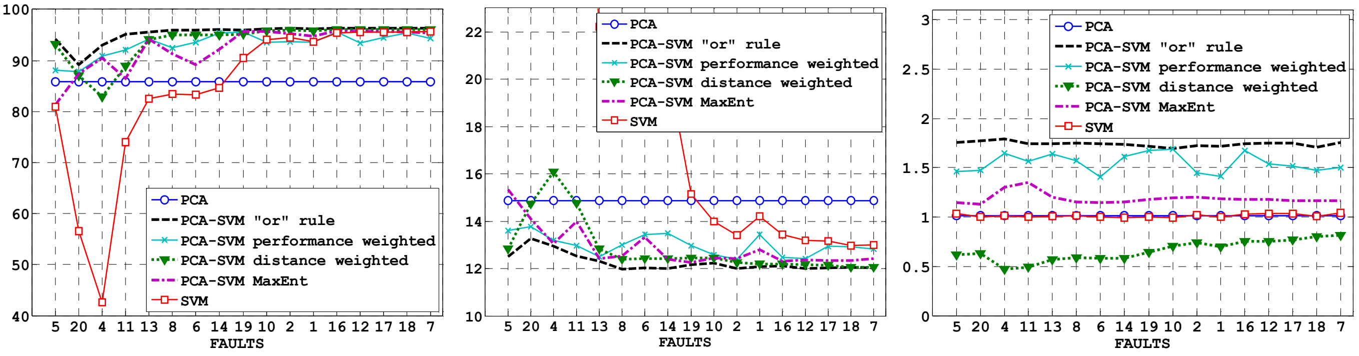
C. *Experimental Results*

In the first experiment we compared accuracies of stand-alone PCA and SVM to joint PCA-SVM model with different fusion strategies, when all training data were labeled. The procedure started with training the PCA model on D_{int}. Then, batches of faulty data were introduced at different plants in this order: 5, 20, 4, 11, 13, 8, 6, 14, 19, 10, 2, 1, 16, 12, 17,18, 7. The SVM and the fusion rule parameters were updated after each fault occurrence. The order in which the faults were introduced was randomly selected. To avoid over-fitting, after each new batch was observed the voting weights were updated before the new supervised model was trained.

Figure 8 shows fault-specific TPR of SVM on test set. It tracks the TPR of each fault type (y-axis) as new fault types were observed (x-axis). Each pie represents TPR (full pie has TPR = 100%, empty pie has TPR = 0%). It can be concluded that some fault types were accurately detected even before they were observed (e.g. SVM trained only on fault 5 had high TPR on faults 1, 2, 4, 6, 7, 8, 12, 13, 14), while for other fault types, the satisfactory FPR was achieved only after they were observed (e.g. faults 10, 11, 16, 19).

Figure 9 reports changes of average TPR, DD and FPR values on test set (y-axis) as new faults were introduced (x-axis). The performance of PCA was the same throughout the procedure, as expected. On the other hand, SVM fault detection performance improved as new fault types were observed. SVM was initially inferior to PCA, but it overtook it after the 10*th* fault type was observed.

It is interesting to observe that while accuracy of SVM after observing only the first fault type (fault 5) was acceptable, it deteriorated dramatically after the second and third fault types (faults 20 and 4) were introduced, and then it recovered after fault types 11 and 13 were observed. The drop in accuracy nicely illustrates the drawbacks of purely supervised fault detectors when only a small fraction of possible fault types are available for training. It can be seen from Figure 8, and it is

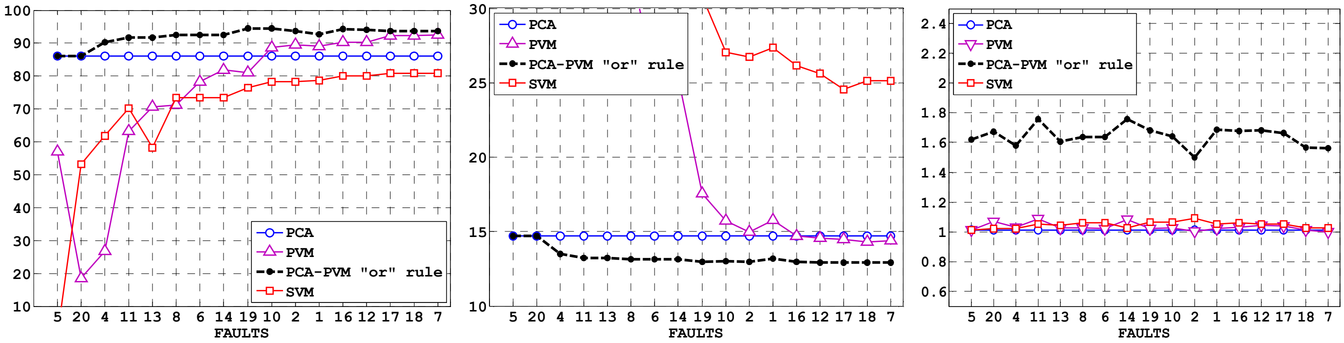


(a) y-axis: TPR (%)

(b) y-axis: Detection Delay (DD)

(c) y-axis: FPR (%)

Fig. 9. Fault detection performance evaluation for PCA, SVM and PCA-SVM models trained using labeled data in terms of: a) TPR b) DD c) FPR



(a) y-axis: TPR (%)

(b) y-axis: Detection Delay (DD)

(c) y-axis: FPR (%)

Fig. 10. Fault detection performance evaluation for PCA, PVM and PCA-PVM models trained using partially labeled data in terms of: a) TPR b) DD c) FPR

consistent with the illustrative example in Figure 5, that while the SVM trained on the first 3 faults achieves high accuracy on these faults, its decision boundary changes in such a way that its accuracy on yet unseen fault types 1, 6, 7, 8, 12, 13, 14, 17, and 18 is low. It is interesting to observe that the corresponding decrease in accuracy of the PCA-SVM models after fault types 5, 20, and 4 were observed was markedly smaller, demonstrating the appeal of the proposed fusion approach for cold start fault detection.

By analyzing Figure 9 further, it can be concluded that PCA-SVM model using different voting strategies achieved better TPR and DD than either PCA or SVM model alone, especially during the cold start period of the experiment (when less than 13 fault types were observed). After the 13*th* fault type was observed, SVM achieved similar accuracy and the detection delay as the fusion model.

We can make several conclusions regarding the FPR. Since the observed "or" rule fusion model FPR was ~1.7%, we can conclude that there was not much overlap between PCA and SVM false alarms. Consequently, distance-weighted voting strategy was able to achieve better FPR performance than individual SVM or PCA, by efficiently solving the disagreement of SVM and PCA via prediction uncertainties.

Judging by the results, the choice of the voting strategy could depend on the specific application at hand. Out of all the strategies discussed in Section IV.A, the "or" rule achieved the highest TPR. The "or" rule also led to the lowest detection delay. However, it fell behind other voting strategies in FPR, where distance-weighted rule was the best overall. In fact, after the 5*th* fault was observed, the distance-weighted PCA-SVM achieved almost the same TPR and DD as the "or" rule PCA-SVM, while having lower FPR. MaxEnt PCA-SVM also fell behind "or" rule in the cold start period, with respect to TPR and DD, but had better TPR.

In the second experiment we evaluated performance of standalone PCA and PVM, and PCA-PVM that used "or" rule, when data batches were partially labeled. We compared them to SVM model that used only labeled data for training. Faults were introduced in the same order as in the previous experiment. Figure 10 shows the comparison on test set in terms of TPR, FPR and DD. SVM was inferior to individual PCA and PVM models, due to small amounts of labeled data available for training. By comparing PCA-PVM to stand-alone PCA and PVM, similar conclusions to the ones in the previous experiment can be drawn. However, the improvement of the fusion model over the individual models had a smaller margin, due to scarcity of labeled data. Nevertheless, the experiment shows that leveraging supervised and unsupervised models' strengths is beneficial even with limited amounts of label data.

VI. Conclusion

In this paper we proposed a cold start fault detection framework applicable to the scenarios in which historical data are labeled or partially labeled. The proposed methodology utilizes advantages of unsupervised and supervised fault detection models by combining their predictions using rules that range from simple "or" rule to more involved Maximum Entropy-based fusion. The supervised model is updated as new fault types are observed. Depending on whether new data are partially or completely labeled, the model was updated in the standard supervised or the proposed semi-supervised manner, respectively. Experimental results on benchmark data indicate that combining models is beneficial, as the fusion model achieves better performance than standalone models during the cold start stage.

References

[1] S. Simani, "Identification and Fault Diagnosis of a Simulated Model of an Industrial Gas Turbine", IEEE Transactions on Industrial Informatics, vol. 1, no. 3, pp. 202-216, 2005.

[2] Y. Zhang, H. Zhou, S. J. Qin, and T. Chai, "Decentralized Fault Diagnosis of Large-Scale Processes Using Multiblock Kernel Partial Least Squares", IEEE Transactions on Industrial Informatics, vol. 6, no. 1, pp. 3-10, 2010.

[3] L. H. Chiang, E. Russell, and R. D. Braatz, "Fault detection and diagnosis in industrial systems", Springer, 2001.

[4] D. D. Downs and E. F. Vogel, "A plant-wide industrial process control problem", *Computers & Chemical Engineering*, vol. 17, issue 3, pp. 245-255, 1993.+

[5] D. P. Filev, R. B. Chinnam, F. T., and P. Baruah, "An Industrial Strength Novelty Detection Framework for Autonomous Equipment Monitoring and Diagnostics", IEEE Transactions on Industrial Informatics, vol. 6, no. 4, pp. 767-779, 2010.

[6] J. F. MacGregor, "Statistical process control of multivariate process", *IFAC Int. Symp. on Advanced Control of Chemical Processes*, pp. 427-435, 1994.

[7] W. Ku, R. H. Storer and C. Georgakis, "Disturbance detection and isolation by dynamic principal component analysis", *Chemometrics and Intelligent Laboratory Systems*, vol. 30, pp. 179-196, 1995.

[8] M.-D. Ma, D. S.-H. Wong, S.-S. Jang, and S.-T. Tseng, "Fault Detection Based on Statistical Multivariate Analysis and Microarray Visualization", IEEE Transactions on Industrial Informatics, vol. 6, no. 1, pp. 18-24, 2010.

[9] J. M Leea, C. K. Yoob and I-B. Leea, “Statistical monitoring of dynamic processes based on dynamic independent component analysis”, *Chemical Engineering Science*, vol. 59 pp. 2995-3006, 2004.

[10] A. Hyvärinen, J. Karhunen, and E. Oja, “Independent Component Analysis”, John Wiley & Sons, 2001.

[11] L. H. Chiang, M. E. Kotanchek and A. K. Kordon, “Fault diagnosis based on Fisher discriminant analysis and support vector machines”, Computers and Chemical Engineering, vol. 28(8), pp. 1389-1401, 2003.

[12] A. Kulkarni, V. K. Jayaraman, and B. D. Kulkarni, “Knowledge incorporated support vector machines to detect faults in Tennessee Eastman Process”, *Computers and Chemical Engineering*, vol. 29 pp. 2128–2133, 2005.

[13] A. Widodo, B.S. Yang and T. Han, ”Combination of independent component analysis and support vector machines for intelligent faults diagnosis of induction motors”, *Expert Systems with Applications*, vol. 32, pp. 299-312, 2007.

[14] C. Bo, X. Qiao, G. Zhang, Y. Bai and S. Zhang “An integrated method of independent component analysis and support vector machines for industry distillation process monitoring”, *Journal of Process Control*, 20, pp. 1133–1140, 2010.

[15] I. Monroya, R. Benitezb, G. Escuderoc and M. Graells, “A semi-supervised approach to fault diagnosis for chemical processes”, *Computers & Chemical Engineering*, vol. 34, pp. 631-642, 2010.

[16] C. C. Hsu, L. S. Chen, “Integrate Independent Component Analysis and Support Vector Machine for Monitoring Non-Gaussian Multivariate Process”, *4th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1-6, 2008.

[17] M. Guo, L. Xie, S. Wang and J. Zhang, “Research on an Integrated ICA-SVM Based Framework for Fault Diagnosis”, *IEEE International Conference on Systems, Man and Cybernetics*, pp. 2710–2715, 2003.

[18] R. Gong, S. H. Huang and T. Chen, “Robust and Efficient Rule Extraction Through Data Summarization and Its Application in Welding Fault Diagnosis”, IEEE Transactions on Industrial Informatics, vol. 4, no. 3, pp. 198-206, 2008.

[19] B. A. Foss, T. A. Johansen, “An integrated approach to on-line fault detection and diagnosis -including artificial neural networks with local basis functions”, IFAC symposium in *On-line fault detection and supervision in the chemical process industries*, pp. 207-213, 1993.

[20] J.-H. Zhou, C. K. Pang, F. L. Lewis, and Z.-W. Zhong, “Intelligent Diagnosis and Prognosis of Tool Wear Using Dominant Feature Identification”, IEEE Transactions on Industrial Informatics, vol. 5, no. 4, pp. 454-465, 2009.

[21] G. Blanchard, G. Lee, C. Scott, Semi-Supervised Novelty Detection, Journal of Machine Learning Research vol. 11, pp. 2973-3009, 2010.

[22] H. Chen, G. Jiang, C. Ungureanu and K. Yoshihira, “Combining supervised and unsupervised monitoring for fault detection in distributed computing systems”, *ACM symposium on Applied computing*, pp. 705–709, 2006.

[23] V.N. Vapnik, *Statistical Learning Theory*, John Wiley Sons, Inc.,1998.

[24] N. Cristianini, J. S.-Taylor, An introduction to support vector machines and other kernel-based learning methods, Cambridge Press 2000.

[25] X. Zhu, Z. Ghahramani, J. Lafferty, “Semisupervised learning using gaussian fields and harmonic functions”, *International Conference on Machine Learning* (ICML), pp. 912–919, 2003.

[26] K. Zhang and J.T. Kwok, “Prototype vector machine for large scale Semi-supervised Learning”, *International Conference on Machine Learning* (ICML), pp. 1233–1240, 2009.

[27] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, B. Scholkopf, “Learning with local and global consistency”. Neural Information Processing Systems, pp. 321–328, 2003.

[28] M. Belkin, P. Niyogi, V. Sindhwani, “Manifold regularization: a geometric framework for learning from labeled and unlabeled examples” *Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, 2006.

[29] K. Ghosh, Y. S. Ng, R. Srinivasan, “Evaluation of decision fusion strategies for effective collaboration among heterogeneous fault diagnostic methods”, *Computers and Chemical Engineering*, vol 35. pp. 342–355, 2011.

[30] A. Ratnaparkhi, “A maximum entropy model for part-of-speech tagging”, *Proc. EMNLP*. New Brunswick, New Jersey: Association for Computational Linguistics, pp. 133-142, 1996.

[31] M Grbovic, W Li, P Xu, AK Usadi, L Song, S Vucetic, " Decentralized fault detection and diagnosis via sparse PCA based decomposition and Maximum Entropy decision fusion", Journal of Process Control, vol. 22, pp. 738–750 2012.

[32] LIBSVM Library, http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Mihajlo Grbovic received his B.A. and M.S. degree in electrical engineering from the Faculty of Technical Sciences, University of Novi Sad in 2007. He is currently pursuing a Ph.D. degree in computer and information sciences at Temple University, Philadelphia, PA, and is expected to graduate in 2012.

His research interests include machine learning, data mining, data-driven fault detection, and computational advertising

Weichang Li (S’04-M’05) received the B.S. and M.S. degrees in acoustics and electronic engineering from Harbin Engineering University in 1993 and 1996, the dual M.S. degree in electrical engineering and computer science and ocean engineering from Massachusetts Institute of Technology, Cambridge, in 2002, and the Ph.D. degree in electrical and oceanographic engineering from the Massachusetts Institute of Technology/Woods Hole Oceanographic Institute Joint Program in Oceanography in February 2006. He is currently with the ExxonMobil Research and Engineering Company, Annandale, NJ. His research interests include statistical signal processing, machine learning, acoustic communications, and 3D imaging. He is also a member of SIAM, SEG and ASA.

Niranjan Subrahmanya received his Bachelors and Masters degrees in Mechanical Engineering from the Indian Institute of Technology, Mumbai in August, 2003 and a PhD degree from Purdue University, West Lafayette, Indiana, in May 2009. He is currently working with the Complex Systems Science group at Exxon Mobil Research and Engineering Company, Annandale, NJ. His research interests include machine learning, signal processing, intelligent systems, dynamics and control, and data-based systems modeling, monitoring and diagnostics. He is a member of IEEE and SIAM.

Adam Usadi received a BS in Physics from Dartmouth College, a MS in materials science from Hiroshima University, and a PhD in Space Physics from Rice University. Adam worked for Goldman Sachs where he supported precious metals trading. At ExxonMobil's Upstream Research Company in Houston he worked on computational math problems in reservoir modeling and seismic processing. At ExxonMobil's Corporate Strategic Research Company in New Jersey he's served as Section Head of the Complex Systems Science section where he led the groups in data analytics as well as supply chain optimization. Currently he is Section Head of the Emerging Energy Science Section.

Slobodan Vucetic received the B.S. and M.S. degrees in electrical engineering from the University of Novi Sad, Serbia, and the Ph.D. degree in electrical engineering from Washington State University, Pullman, in 1994, 1997, and 2001, respectively.

He is currently an Associate Professor in the Department of Computer and Information Sciences, Temple University, Philadelphia, PA. His research interests are data mining and machine learning