Kernel Classifier with Adaptive Structure and Fixed Memory for Process Diagnosis

Haiqing Wang and Ping Li
National Lab of Industrial Control Technology, Zhejiang University, Hangzhou, 310027, China

Furong Gao
Dept. of Chemical Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

Zhihuan Song
National Lab of Industrial Control Technology, Zhejiang University, Hangzhou, 310027, China

Steven X. Ding
Institute of Automatic Control and Computer Systems, University of Duisburg-Essen, 47057 Duisburg, Germany

DOI 10.1002/aic.10982
Published online September 1, 2006 in Wiley InterScience (www.interscience.wiley.com).

A unified least-square kernel (ULK) framework is formulated for both process modeling and fault diagnosis issues. As a specific algorithmic implementation of the ULK method, an adaptive kernel learning (AKL) network classifier is developed for process diagnosis, which is a two-stage online learning algorithm with a fixed-memory strategy. A new concept of space angle index is proposed to structure the growth of node and actively control the complexity of the network. The AKL network performs a backward decreasing for forgetting an old pattern and a forward increasing for incorporating a new online pattern. The recursive algorithms for both stages are derived for quick online updating. Applications of the AKL network to two numerical cases and the Tennessee Eastman process show good performance in comparison to other established methods, and new insights on the pattern recognition for fault diagnosis arising from this flexible classifier are highlighted. © 2006 American Institute of Chemical Engineers *AIChE J,* 52: 3515–3531, 2006

Introduction

Fault diagnosis for industrial process has attracted much attention of researchers from many fields. Several survey papers and monographs on process fault monitoring and diagnosis have been published, highlighting the significance of the subject area for the industry.[1-4] A chemical process is characterized by nature as large scale, nonlinear, and complex with usually time-varying kinetics. Because of this, among the vast fault diagnosis literature only a few methods, especially those driven by *data* or *knowledge*, are relatively more suitable for industrial applications.

Specifically, multivariate statistical process monitoring (SPM) methods—such as principal component analysis (PCA) and partial least squares (PLS)—have been widely applied for chemical process monitoring in the last decade. Various extensions to these SPM methods were proposed and applied.[5-9] However, most of those methods use only the nominal operating data for modeling, and thus other diagnostic tools,[3,8] such as contribution plot and fault subspace, are applied for fault diagnosis; further, SPM methods have a strong statistical requirement and linearity assumption, greatly limiting their applications.

H. Wang is also affiliated with Institute of Automatic Control and Computer Systems, University of Duisburg-Essen, Bismarckstr. 81, BB-511, 47057 Duisburg, Germany.
Correspondence concerning this article should be addressed to H. Q. Wang at hqwang@iipc.zju.edu.cn and S. X. Ding at s.x.ding@uni-duisburg.de.

Other types of data/knowledge-driven process diagnosis approaches include, but are not limited to, neural networks, pattern recognition, fuzzy theory, and the digraph method.[10-13] Because the fault information can generally be used in a supervised manner, these methods can incorporate fault monitoring and diagnosis together. However, this type of methods is typically problem specified and requires that the designer be fairly familiar with the flow sheet of a process.

Recently, statistical learning and kernel (SLK) theory and the support vector machine (SVM) algorithm have found increasing numbers of applications in the chemical industry.[16] SLK theory solves the learning problem with a small sample set in a rigorous mathematical manner, and its theoretical foundation has been laid in the second half of the twentieth century by such pioneers as Vapnik and Chervonenkis (VC theory, optimal separating hyperplane, capacity measures), Aronszajn [reproducing kernel Hilbert spaces (RKHS)], Tikhonov and Arsenin (regularization and ill-posed problems), Wahba (regularization in RKHS), and many others.[14,15,17,18] Nowadays the kernel methods, which can be understood as the algorithmic implementation of statistical learning theory, involving SVM, regularization network, boosting, and Gaussian processes, have been successfully applied to many various fields, such as text categorization, gene expression analysis, and DNA and protein analysis. A number of good references and books on this subject can be found,[15,17-20] and a brief review on the preliminaries necessary for process fault diagnosis will be given in the next section.

Application of SLK theory and SVM in the chemical industry now is still limited, although its increasing growth in the near future is foreseeable. Desai et al.[21] applied the support vector regression (SVR) method to build soft sensors for fedbatch processes. Their comparison study shows that the SVR soft sensor outperforms the multilayer radial basis neural networks generally; Yan et al.[22] also demonstrated that an optimized SVM soft sensor estimating the freezing point of light diesel oil in a distillation column performs better than a neural network based strategy; Lee et al.[23] successfully obtained some quality predictions for polymerization processes with a weighted SVR method.

In addition to the modeling applications, SVM based methods have also been successfully applied to solving multiclass pattern recognition problems in the chemical industry such as operation state differentiation and process fault diagnosis. Trafalis et al.[24] used the SVM classifier to differentiate the vertical and horizontal two-phase flow regimes in a pipe so as to predict the transition region between the flow regimes; Jemwa and Aldrich[25] developed a decision support system by integrating a soft margin SVM classifier with decision tree technique; Chiang et al.[26] compared the performances of FDA, SVM, and proximal SVM for fault diagnosis of the Tennessee Eastman (TE) process. It was found that it is important to properly select input variables and time lag to train the classifier.

Interestingly, although both the modeling and diagnosis can be considered as the *same* general learning problem,[15,17] these two issues are actually solved by using *different* kernel algorithms. In this article, a new multiclass classification scheme for process fault diagnosis is developed from the SLK theory and SVM algorithm, resulting in a unified solution to both process monitoring and diagnosis. That is, a single kernel algorithm can be applied simultaneously to these two learning tasks.

The remainder of this article is so organized. In Section 2, a unified least-square kernel (ULK) framework is developed after a brief introduction to SLK theory and SVM. This proposed framework provides the possibility of a unified algorithm for process modeling (regression) and diagnosis (classification) issues. In Section 3, an adaptive kernel learning (AKL) network is developed together with a new concept of space angle index (SAI) for node growth determination. Applications of the proposed AKL network are illustrated in Section 4 by two numerical classification problems and the Tennessee Eastman process. The conclusions are drawn in the final section.

Least-Square Kernel Learning and Its Unified Form

Fault diagnosis (or classification in the field of SLK) for an industrial process can be considered as learning a mapping function f: $\mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X} \in \mathbb{R}^M$ and $\mathcal{Y} \in \mathbb{R}^N$ are samples/patterns from a data set $L = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\} \subset \mathcal{X} \times \mathcal{Y}$, consisting of ℓ independent identically distributed (iid) samples drawn from the probability distribution on $\mathcal{X} \times \mathcal{Y}$. Output vector set $\mathcal{Y}$ usually takes only $\{\pm 1\}$ as its elements for diagnosis issue. The joint and conditional probabilities over $\mathcal{X}$ and $\mathcal{Y}$ obey the following relation: $p(x, y) = p(y \mid x)p(x)$, which is assumed as fixed but unknown.

This formulation is rather general, which may also describe other problems such as model identification/regression and density estimation.[15,17,18] The difference depends only on the specific form of *cost function* chosen for different problems. Thus there is a close connection between modeling and fault diagnosis and we will address this relation in later sections. The SVM as one of the earliest machine learning algorithms will be briefly reviewed as well. Because SVM itself has many different variants; in this article we limit the presentation to the basic SVM classifier using soft margin cost function.

Preliminaries for SLK and SVM

A good fault diagnosis system should have the ability of generalizing to unseen future process patterns with a small risk

$$R[f] = \int_{\mathcal{X}\times\mathcal{Y}} c[x, y, f(x)]dp(x, y) \tag{1}$$

where c is a loss function. In the case of fault diagnosis, a common choice is $c[x, y, f(x)] = (1/2)|f(x) - y|$, or soft margin loss function $c[x, y, f(x)] = \max[0, 1 - yf(x)]$; for process modeling, however, the square loss function $c[x, y, f(x)] = [y - f(x)]^2$ is more prevailing.

The above risk cannot be evaluated because the distribution p is unknown. An alternative to infer function f from the training set L is based on the minimization of *empirical risk*

$$R_{emp}[f] = \frac{1}{\ell}\sum_{i=1}^{\ell} c[x_i, y_i, f(x_i)] \tag{2}$$

The direct minimization of Eq. 2 is an *ill-posed* problem,[17,18] if no restrictions on the function class from which the decision function f is estimated. Machine learning theory studies the

way of implementing such restrictions, by taking into account the capacity or complexity measured by such terms as the VC dimension. The VC theory provides bounds on the test error, and the minimization of these bounds, which depends on both the empirical risk and the capacity of the function class, leads to the principle of *structural risk minimization.*[17,18]

Assume the decision function $f \varepsilon \mathcal{H}$, which is a Hilbert space endowed with a dot product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Generally, it can be further assumed that $\mathcal{H}$ is a RKHS.[18] As a temporal step of the kernel based learning method, one can consider mapping the input pattern/sample $x_i \varepsilon \mathcal{X}$ first into $\mathcal{H}$ (RKHS *feature space*), by ϕ: $\mathcal{X} \rightarrow \mathcal{H}$, where ϕ is a nonlinear operator associated with a *positive definite* kernel k. The dimension of space $\mathcal{H}$ is much higher than that of the original $\mathcal{X}$ and can even be infinite, such as in the case of a Gaussian kernel.[18] Functional f is then determined in some optimal sense to ascertain the decision function/the regression model desired.

The standard soft margin SVM binary classifier is the solution of the following optimization problem:

$$\min_{\mathbf{w}\in\mathcal{H},b\in R} \frac{1}{2}\|\mathbf{w}\|^2 + \lambda \sum_{i=1}^{\ell} \xi_i^2 \tag{3}$$

$$s.t. \begin{cases} y_i[\langle \mathbf{w}, \phi(x_i)\rangle + b] \geq 1 - \xi_i \\ \xi_i \geq 0, \lambda > 0, i \in [1, \ell] \end{cases}$$

where the regularization constant λ determines the trade-off between margin maximization and training error minimization; slack variables ξ_i are used to relax the constraints of hard margin SVM classifier so that a few of the misclassified data are permitted; the solution, that is, weighting vector $\mathbf{w}$ and bias term b, gives the corresponding decision function as

$$f(x) = \text{sgn}[\langle \mathbf{w}, \phi(x_i)\rangle + b] \tag{4}$$

More specifically, the solution $f(x)$ is actually "supported" by limited expanding terms that correspond to a small subset of the train data set L, that is, *support vectors* (SVs).

By suitably choosing another cost function, just as previously mentioned, the solution to this general statistical learning problem of Eq. 2 turns to the *process modeling* problem. In this case the cost function and its constrains are expressed as

$$\min_{\mathbf{w}\in\mathcal{H},b\in R} \frac{1}{2}\|\mathbf{w}\|^2 + \lambda \sum_{i=1}^{\ell} (\xi_i + \xi_i^*) \tag{5}$$

$$s.t. \begin{cases} y_i - \langle \mathbf{w}, \phi(x_i)\rangle - b \leq \varepsilon + \xi_i \\ \langle \mathbf{w}, \phi(x_i)\rangle + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i > 0, \xi_i^* > 0 \end{cases}$$

where the constant ε is a tolerable threshold and both ξ_i and ξ_i^* are slack variables. The roles of other variables are identical to those in Eq. 3.

Equations 3 and 5 are presented here without proving and solution, for which details can be found in some basic literature on SLK theory.[18-20] Although the classification and regression can be formulated as the same general statistical learning issue, they are actually solved separately using different cost function forms, leading to different algorithms. Besides this, the current SVM-based learning methods suffer from several other disadvantages for industrial applications:

(1) The training of the most SVM-based algorithms is in a batch manner, although the obtained learning machine runs online for testing.

(2) The solution of Eq. 3 or Eq. 5 is a quadratic planning (QP) problem, which may be computationally heavy for a long-term monitoring task. Thus, online solution of the fault diagnosis problem is difficult considering its large computation load.

(3) Storing and calculating of the large *Kernel Matrix* is problematic for industrial applications where the amount of process data increases with increasing time.

Unified least-square kernel method

The source of the QP problem in the basic SVM arises from inequality constraints. Several researchers independently proposed using the equality rather than inequality constraints for the new SVM. Suykens et al.[19] proposed the LS-SVM, which retains all the other features of traditional SVM, including using different forms of cost functions for classification and regression. Fung and Mangasarian[27] developed a proximal SVM classifier, which classifies samples to the closer one of two parallel planes, rather than to one of two disjoint half-spaces as the standard SVM. Rifkin et al.[28] also proposed adopting equality constraints and recommended a Tikhonov regularization term, where a square loss function $[y - f(x)]^2$ was chosen empirically for classification issue without much elaboration. This is a natural choice for regression problems, but it seems odd for the classification problem at the first glance; in the positive class, for example, large positive predictions are almost as costly as large negative predictions.

In fact, it can be easily shown that the common classification loss $[1 - yf(x)]^2$ can be rewritten as

$$[1 - yf(x)]^2 = [yy - yf(x)]^2 = [y - f(x)]^2 \tag{6}$$

where $y = \pm 1$.

A relation similar to Eq. 6 has been explored in Suykens et al.[19] and Allwein,[29] although none of them aimed to unify and solve the classification and regression problems using a single algorithm. In the following part, a more general framework, that is, the ULK method, will be first presented and then solved by our proposed AKL algorithm.

To this end, an optimization problem, adopting the square loss function and Tikhonov regularization with *time* and *output channel* indices, is proposed:

$$\min_{f\in\mathcal{H}} \frac{1}{2}\|\boldsymbol{\varepsilon}_{t,p}\|^2 + \lambda\Omega[\|f_{t,p}\|_{\mathcal{H}}] \tag{7}$$

$$st: \begin{cases} y_{t,p} - f_p(\mathbf{x}_t) - \boldsymbol{\varepsilon}_{t,p} = 0 \\ \lambda \geq 0, p \in [1, P], t \in [1, \ell] \end{cases}$$

where, for the pth output channel and at time t, $\mathbf{x}_t = [x_{t,1}, x_{t,2}, \ldots, x_{t,M}]^T$ is the input data vector, $y_{t,p}$ is the output scalar, $f_{t,p} = f_p(\mathbf{x}_t)$ is the learning function, and $\boldsymbol{\varepsilon}_{t,p} = [\boldsymbol{\varepsilon}_{1,p}, \boldsymbol{\varepsilon}_{2,p}, \ldots, \boldsymbol{\varepsilon}_{t,p}]^T$ is the approximation error; λ is the regularization param-

eter to control the smoothness of a solution; and $\Omega[\|f_{t,p}\|]$ is the regularization term, which is usually chosen to be convex.

According to the *Representer Theorem* in RKHS,[18] it can be known a priori that the solution to Eq. 7 has the following SV expansion form:

$$f_{t,p} = \sum_{i=1}^{t} \beta_{t,p} k(\mathbf{x}_i, \mathbf{x}_t) \tag{8}$$

where $\boldsymbol{\beta}_{t,p} = [\beta_{1,p}, \beta_{2,p}, \ldots, \beta_{t,p}]^T$ is the weighting vector of the pth output channel until time t.

By setting $\Omega[\|f\|] \doteq (1/2)\|\boldsymbol{\beta}_{t,p}\|^2$ and substituting Eq. 8 into Eq. 7, the proposed unified optimization problem for both the *multiclass classification* and *MIMO regression*, can be simply formulated as an unconstrained version in space $\mathbb{R}^t$

$$\min_{\boldsymbol{\beta}_{t,p} \in \mathbb{R}^t} \frac{1}{2}\|\mathbf{y}_{t,p} - \mathbf{K}\boldsymbol{\beta}_{t,p}\|^2 + \frac{\lambda}{2}\|\boldsymbol{\beta}_{t,p}\|^2 \tag{9}$$

where parameters $\lambda \geq 0$, $p \in [1, P]$, $t \in [1, \ell]$; $\mathbf{y}_{t,p} = [y_{1,p}, y_{2,p}, \ldots, \mathbf{y}_{t,p}]^T$ is the output vector of the pth channel until time t.

The same formulation as Eq. 9 was proposed by Wang et al. for MIMO identification, from a different methodological perspective as the general "kernelized" form of MIMO linear model using the *kernel trick*.[30] For the case of fault diagnosis/classification, a decision function with a form similar to Eq. 4 is adopted.

The dimension of the solution to Eq. 9 now is limited (although it will increase linearly with time t); thus, the gradient decreasing method can be applied directly in the primal space and no conversion to a dual problem after which it can be solved, as in the case of traditional SVM or LS-SVM method.[18-20]

The other issue of concern is the selection of coding/decoding schemes. It was reported that different coding/decoding schemes for SVM result in different classification performances.[29,31] Derivation of the proposed ULK problem is independent of the coding/decoding scheme adopted. In this article, a one-vs.-others strategy[18] is adopted because of its practicability and computational efficiency. However, it should be noted that, because a special sparsity strategy (in the next section) is used, Eq. 7, as well as Eq. 9, will give a *multiclass classifier* but not *multiple binary classifiers*.

Adaptive Kernel Learning Network

The *AKL network* is a specific algorithmic implementation of the above-formulated ULK framework. One of its characteristics is that a *pre-sparsity* step is performed *before* solving the learning machine. This makes the complexity of the learning machine under control and online recursive learning possible.[30] On the other hand, the time-varying property of industrial process requires that the memory of the learning machine is limited, that is, it should be fed with fixed-length data. This issue constitutes another main characteristic of the proposed AKL network.

Dependency detection and network growth

The length of parameter vector $\boldsymbol{\beta}_{t,p}$, which can be loosely understood as the order (or an indicative complexity measure) of the obtained learning machine, equals the number of elements in L. With continuous online learning (in classification or regression), the model complexity will increase steadily, which makes the computation impracticable. This article resolves this problem by exploring the linear dependency of samples in feature space $\mathcal{H}$, by introducing the concept of *space angle index* (SAI). This provides an *incremental* manner to find the "nodes" of the proposed AKL network (which can be conceived as a kind of "SVs," similar to the solutions of Eqs. 3 and 5). This network-like formulation of the proposed algorithm is natural because it is well known that the SVM has a single layer network topology.[18-20]

The *Node Set* of AKL network at time i is denoted as $N_i = \{\phi(\tilde{x}_1), \ldots, \phi(\tilde{x}_{n_i})\}$, $i = 1, \ldots, t$, and satisfying $n_i \leq t$. The subspace spanned by the vectors in set N_i is denoted as $F_i \subset \mathcal{H}$. At the beginning of learning, there is only one feature vector in the node set, that is, $N_1 = \{\phi(\tilde{x}_1)\} = \{\phi(x_1)\}$. The new feature vectors will be added into the set according to their dependencies on the space F_i using the following proposed SAI criterion.

Define the sine value between the tth input feature vector $\phi(x_t)$ and the node space F_{t-1} in $\mathcal{H}$ as *SAI*

$$\sin(\theta_t) = \min_{\mathbf{a}_t} \left\| \phi(x_t) - \sum_{k=1}^{n_t - 1} a_k \boldsymbol{\phi}(\tilde{x}_k) \right\| \Big/ \|\boldsymbol{\phi}(x_t)\| \tag{10}$$

where $\mathbf{a}_t = [a_1, \ldots, a_{n_t - 1}]^T$ is the corresponding linear combination coefficient vector.

If $\sin(\theta_t) > \sqrt{v_0}$, where v_0 is a predefined small value satisfying $0 \leq \sqrt{v_0} \leq 1$, then the vector $\phi(x_t)$ will be introduced to constitute the expanded node set, $N_t = \{N_{t-1}, \phi(x_t)\}$; otherwise, this vector will be rejected without the node set expansion, that is, $N_t = N_{t-1}$. Note the minimization of Eq. 10 is concerned only with its numerator item; its square function is defined as

$$\varphi_t = \min_{\mathbf{a}_t} \|\boldsymbol{\phi}(x_t) - \tilde{\boldsymbol{\Phi}}_{t-1}\mathbf{a}_t\|^2 = \min_{\mathbf{a}_t} \{k_{tt} - 2\mathbf{a}_t^T \tilde{\mathbf{k}}_t + \mathbf{a}_t^T \tilde{\mathbf{K}}_{t-1}\mathbf{a}_t\} \tag{11}$$

where $\tilde{\boldsymbol{\Phi}}_{t-1} = [\phi(\tilde{x}_1), \ldots, \phi(\tilde{x}_{n_t - 1})]$ is the *node matrix* at time $t - 1$. The quantities of $k_{tt} = \langle \phi(x_t), \phi(x_t) \rangle$, $\tilde{\mathbf{k}}_t(i, t) = \langle \phi(\tilde{x}_i), \phi(x_t) \rangle$, and $\tilde{\mathbf{K}}_{t-1} = \langle \phi(\tilde{x}_i), \phi(\tilde{x}_j) \rangle$, $i = 1, \ldots, n_t - 1$, are all corresponding kernel transforms, as in Eq. 8.

The minimization of Eq. 11 is straightforward and yields

$$\begin{cases} \varphi_t = |k_{tt} - \mathbf{a}_t^T \tilde{\mathbf{k}}_t| \\ \mathbf{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_t \end{cases} \tag{12}$$

Therefore, by applying the SAI as Eq. 10 actually equals—to judge the following simple condition—

$$\varphi_t / k_{tt} \leq v_0 \tag{13}$$

The geographical illustration of the above node selection is shown as Figure 1. In the case of a Gaussian kernel, the length of $\|\phi(x)\|$, that is, the denominator of Eq. 10 will equal 1. The idea of using SAI to represent data in lower dimensional space is similar to that of principal component analysis (PCA).[8]

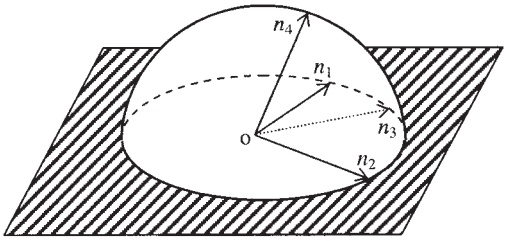


Figure 1. Node selection of AKL network using Gaussian kernel (unit circle) in $\mathcal{H}$.

Samples n_1 and n_2 are initial nodes; sample n_3 will not be selected as a node, but n_4 will be.

Different from PCA, the nodes (that is, basis vectors) for data representation are selected directly from the training set, but not their linear combinations.

Online calculation of SAI using Eq. 13 allows the node set to adaptively learn new process dynamics by adding new "nodes" into it. At the same time, the inputs in the feature space are greatly compressed by using the node set:

$$\mathbf{\Phi}_t = \tilde{\mathbf{\Phi}}_t[\mathbf{a}_1, \ldots, \mathbf{a}_t] = \tilde{\mathbf{\Phi}}_t \mathbf{A}_t^T \tag{14}$$

where the elements of *transform matrix* $\mathbf{A}_t$ are defined as $\mathbf{A}_t(i, j) = a_{i,j}$ as in Eq. 12 for $i \le j$, and equal to 0 for $i \ge j$.

Using the node set representation of Eq. 14, the cost function of Eq. 9 changes to a "reduced" form

$$J(\tilde{\boldsymbol{\beta}}_{t,p}) = \frac{1}{2}\|\mathbf{y}_{t,p} - \mathbf{A}_t\tilde{\mathbf{K}}_t\tilde{\boldsymbol{\beta}}_{t,p}\|^2 + \frac{\lambda}{2}\|\tilde{\boldsymbol{\beta}}_{t,p}\|^2 \tag{15}$$

where $\tilde{\boldsymbol{\beta}}_{t,p} = \mathbf{A}_t^T\boldsymbol{\beta}_{t,p}$.

Note the length of $\tilde{\boldsymbol{\beta}}_{t,p}$, much shorter than that of $\boldsymbol{\beta}_{t,p}$, equals the number of nodes at time t. The solution to Eq. 15 is straightforward and can be obtained by differentiating $J(\tilde{\boldsymbol{\beta}}_{t,p})$ respective to $\tilde{\boldsymbol{\beta}}_{t,p}$:

$$\partial J(\tilde{\boldsymbol{\beta}}_{t,p})/\partial\tilde{\boldsymbol{\beta}}_{t,p} = (\mathbf{A}_t\tilde{\mathbf{K}}_t)^T(\mathbf{A}_t\tilde{\mathbf{K}}_t\tilde{\boldsymbol{\beta}}_{t,p} - \mathbf{y}_{t,p}) + \lambda\tilde{\boldsymbol{\beta}}_{t,p} = 0 \tag{16}$$

and the model coefficient vector is obtained as ($p = 1, \ldots, P$)

$$\tilde{\boldsymbol{\beta}}_{t,p} = [(\lambda\mathbf{I} + \tilde{\mathbf{K}}_t\mathbf{A}_t^T\mathbf{A}_t\tilde{\mathbf{K}}_t)^{-1}\tilde{\mathbf{K}}_t\mathbf{A}_t^T]\mathbf{y}_{t,p} \tag{17}$$

It should be noted that the first part of the above equation (in square brackets) concerns the input space $\mathcal{X}$ and it is shared by all output channels. This is an important property of the algorithm because the dominating computational burden of the proposed AKL network algorithm lies with this part.

The study of dependency/multicollinearity of mapped features is far from new. The geometric formulation of proposed SAI has an advantage of physical intuition, and the number of nodes in the AKL network can be adjusted by parameter v_0 in Eq. 13, similar to v-SVM.[18] Several existing *pre-sparsity* approaches[32,33,41] may be considered as special cases of Eq. 13 when a Gaussian kernel is used.

Another kind of sparsity approaches is performed *after* the learning procedure is completed; we refer to this type of sparsity approaches as *post-sparsity*. The main motivation of *post-sparsity* is to increase the speed/efficiency of later testing, whereas the *pre-sparsity* is to control the complexity of the learning machine. Schölkopf et al. presented a "reduced set" method by approximating the images of SVs in $\mathcal{H}$ using a (time-consuming) iterative algorithm.[34] However, the obtained reduced set will not be a subset of the original SVs. The other common post-sparsity trick is based on the Nyström method, which randomly selects a few columns/rows of the kernel matrix $\mathbf{K}$ according to a preset distribution to form a lower-rank approximation to the original large $\mathbf{K}$.[35]

Online kernel learning

Although kernel methods have proven to be successful in many *batch* settings, its extension to online form is not straightforward.[39] Even some online (or incremental) learning methods available in the literature cannot also be directly applied to the chemical process because of the limitations outlined previously.

Incremental update algorithms,[36,37] which use some specially designed optimizing procedures to meet the Karush–Kuhn–Tucker (KKT) condition, can solve the SVM cost function incrementally and update the corresponding SVs with reduced computational load compared with the batch optimi-

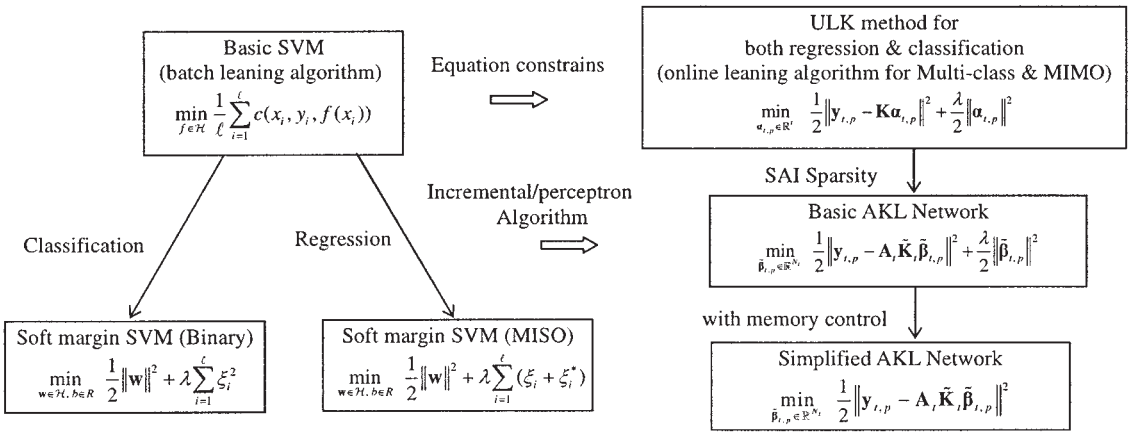


Figure 2. Relations and differences between SVM method and ULK method.

zation. However, this kind of incremental update algorithms requires iteration operations during optimization and cannot guarantee a bound on the number of iterations required per updating.

Recently, several perceptron-like algorithms[38,39] have been proposed for performing updates at each step. Because this kind of learning algorithms is asymptotically convergent in nature, one cannot guarantee that the updating operation can track the process dynamics quickly. Thus, some impractical assumptions are typically required for these algorithms, such as noise-free, nonmoving targets, or an upper bound on the complexity of the estimators.

Wang et al.[30] applied the AKL network to online MIMO identification issue and a recursive learning algorithm for Eqs. 15 and 17 (with $\lambda \neq 0$) has been derived. In the following this algorithm is refined and expanded to process diagnosis issue in a unified form.

Specifically, we will simplify the basic cost function of the AKL network by removing the explicit regularizing part in Eq. 15, which yields .0.0

$$\min_{\tilde{\boldsymbol{\beta}}_{t,p}\in\mathbb{R}^{N_t}} \frac{1}{2}\|\mathbf{y}_{t,p} - \mathbf{A}_t\tilde{\mathbf{K}}_t\tilde{\boldsymbol{\beta}}_{t,p}\|^2 \qquad \text{with } p \in [1, P] \tag{18}$$

Because those input vectors in the linear span of the nodes will be excluded from the set N_t, this ensures that the kernel matrix $\tilde{\mathbf{K}}_t$ is invertible and not subject to the numerical problem. The solution to Eq. 18 can be obtained directly from Eq. 17 by setting $\lambda = 0$:

$$\tilde{\boldsymbol{\beta}}_{t,p} = [\tilde{\mathbf{K}}_t^{-1}(\mathbf{A}_t^T\mathbf{A}_t)^{-1}\mathbf{A}_t^T]\mathbf{y}_{t,p} \tag{19}$$

where $p \in [1, P]$.

The reasons for removing the regularization term are twofold. One is to avoid the possible instability caused by an inappropriately selected value of λ, given that the AKL network is operated online and the rigorous guide for setting this optimal value is complicated and not yet available; the other pertains to the fact that the complexity of the AKL network has been controlled before the learning to some degree by the proposed pre-sparsity procedure.

Figure 2 summarizes the relation and difference of the proposed ULK-based method and the standard SVM-based method.

Backward decreasing learning stage

The aim of backward learning is to recursively delete the old information. Assume at time t the node growth of the AKL network is finished and refer to its corresponding transform matrix as $\mathbf{A}_t \in \mathbb{R}^{N\times n_t}$ and kernel matrix as $\tilde{\mathbf{K}}_t$. This means the structure of the AKL network stops growing and $\tilde{\mathbf{K}}_t$ is fixed during the backward learning procedure.

Let the symbol $N \in \mathbb{R}^+$ as the memory length and we can specify the transform matrix $\mathbf{A}_t$ and model output vector $\mathbf{y}_{t,p}$ further as

$$\mathbf{A}_t = [\mathbf{a}_{t-N+1}, \mathbf{A}_o^T]^T \qquad \mathbf{y}_{t,p}^T = [y_{t-N+1,p}, \mathbf{y}_{o,p}^T]^T \tag{20}$$

where $\mathbf{A}_o = [\mathbf{a}_{t-N+2}, \ldots, \mathbf{a}_t]^T \in \mathbb{R}^{(N-1)\times n_t}$ and the quantity n_t is the number of nodes in the set N_t; additionally, $\mathbf{y}_{o,p} = [y_{t-N+2,p}, \ldots, y_{t,p}]^T \in \mathbb{R}^{(N-1)\times 1}$. In the following, the "o" relates to quantities denoted as shared by their corresponding terms that lie between time t and $t + 1$.

Denote the related quantities respectively as $\mathbf{P}_o = \mathbf{A}_o^T\mathbf{A}_o$, $\mathbf{P}_t = \mathbf{A}_t^T\mathbf{A}_t$, and $\mathbf{A}^- = \mathbf{a}_{t-N+1}\mathbf{a}_{t-N+1}^T$, a recursive relation between matrices $\mathbf{P}_o$ and $\mathbf{P}_t$ can be obtained as

$$\mathbf{P}_o = \mathbf{P}_t - \mathbf{A}^- \tag{21}$$

Applying the Sherman–Morrison–Woodbury formula[40] to Eq. 21 yields the following inverse relation between matrices $\mathbf{P}_o$ and $\mathbf{P}_t$:

$$\mathbf{P}_o^{-1} = \mathbf{P}_t^{-1} + r^- \cdot \mathbf{P}_t^{-1}\mathbf{A}^-\mathbf{P}_t^{-1} \tag{22}$$

where $r^- = 1/(1 - \mathbf{a}_{t-N+1}^T\mathbf{P}_t^{-1}\mathbf{a}_{t-N+1}) \in \mathbb{R}^+$ is a scalar.

By substituting the expressions of Eqs. 20 and 22 into Eq. 19, we arrive at the following *backward* weight vector $\tilde{\boldsymbol{\beta}}_{o,p}$ of simplified AKL network, in a recursive form respect to $\tilde{\boldsymbol{\beta}}_{t,p}$

$$\begin{aligned}
\tilde{\boldsymbol{\beta}}_{o,p} &= \tilde{\mathbf{K}}_t^{-1}\mathbf{P}_o^{-1}\mathbf{A}_o^T\mathbf{y}_{o,p} = \tilde{\mathbf{K}}_t^{-1}[\mathbf{P}_t^{-1} + r^- \cdot \mathbf{P}_t^{-1}\mathbf{A}^-\mathbf{P}_t^{-1}][\mathbf{A}_t^T\mathbf{y}_{t,p} \\
&\quad - a_{t-N+1}y_{t-N+1}] = \tilde{\mathbf{K}}_t^{-1}\mathbf{P}_t^{-1}\mathbf{A}_t^T\mathbf{y}_{t,p} - \tilde{\mathbf{K}}_t^{-1}\mathbf{P}_t^{-1}a_{t-N+1}y_{t-N+1} \\
&\quad + r^- \cdot \tilde{\mathbf{K}}_t^{-1}\mathbf{P}_t^{-1}\mathbf{A}^-\mathbf{P}_t^{-1}[\mathbf{A}_t^T\mathbf{y}_{t,p} - a_{t-N+1}y_{t-N+1}] = \tilde{\boldsymbol{\beta}}_{t,p} - \tilde{\mathbf{K}}_t^{-1}[(\mathbf{I} \\
&\quad + r^- \cdot \mathbf{P}_t^{-1}\mathbf{A}^-)\mathbf{P}_t^{-1}a_{t-N+1}y_{t-N+1} - r^- \cdot \mathbf{P}_t^{-1}\mathbf{A}^-\tilde{\mathbf{K}}_t\tilde{\boldsymbol{\beta}}_{t,p}] = \tilde{\boldsymbol{\beta}}_{t,p} \\
&\quad - \tilde{\mathbf{K}}_t^{-1}(\mathbf{P}_o^{-1}a_{t-N+1}y_{t-N+1} - r^- \cdot \mathbf{P}_t^{-1}\mathbf{A}^-\tilde{\mathbf{K}}_t\tilde{\boldsymbol{\beta}}_{t,p}) = \tilde{\boldsymbol{\beta}}_{t,p} \\
&\quad - \tilde{\mathbf{K}}_t^{-1}\mathbf{s}^-(y_{t-N+1} - \mathbf{a}_{t-N+1}^T\tilde{\mathbf{K}}_t\tilde{\boldsymbol{\beta}}_{t,p})
\end{aligned} \tag{23}$$

where $p \in [1, P]$ and $\mathbf{s}^- = \mathbf{P}_o^{-1}a_{t-N+1}$ is a column vector. The final equation arises from the relation

$$\begin{aligned}
\mathbf{P}_o^{-1}a_{t-N+1} &= \mathbf{P}_t^{-1}a_{t-N+1} + r^- \cdot \mathbf{P}_t^{-1}\mathbf{A}^-\mathbf{P}_t^{-1}a_{t-N+1} = \mathbf{P}_t^{-1}a_{t-N+1} \\
&\quad + \mathbf{P}_t^{-1}a_{t-N+1} \cdot r^-(1 - 1/r^-) = r^- \cdot \mathbf{P}_t^{-1}a_{t-N+1}
\end{aligned}$$

The backward decreasing learning stage will be started once the process data length becomes larger than the memory length, that is, $\ell > N$. To delete the old information sequentially, the corresponding terms $\mathbf{A}_t \in \mathbb{R}^{N\times n_t}$ and $\mathbf{y}_{t,p} \in \mathbb{R}^{N\times 1}$ should be saved. This altogether requires a storage space of $\mathbb{O}[N \times (n_t + 1)]$ for each output channel where the node number n_t is generally small, whereas for batch SVM learning it requires a storage space of $\mathbb{O}(\ell^2)$ for each output channel; moreover, it increases in a quadratic fashion with time, which could present a computational problem for a long operation time with $\ell \gg N$.

Forward increasing learning stage

Different from the backward learning, this learning stage is to grow the nodes with new incoming process information. The forward learning algorithm involves two situations and will be addressed separately.

Scenario 1: Stable Network Structure. After the dominant process dynamics have been learned by the proposed AKL algorithm, the AKL network will become relatively "stable," being characterized as no node growth with the new incoming

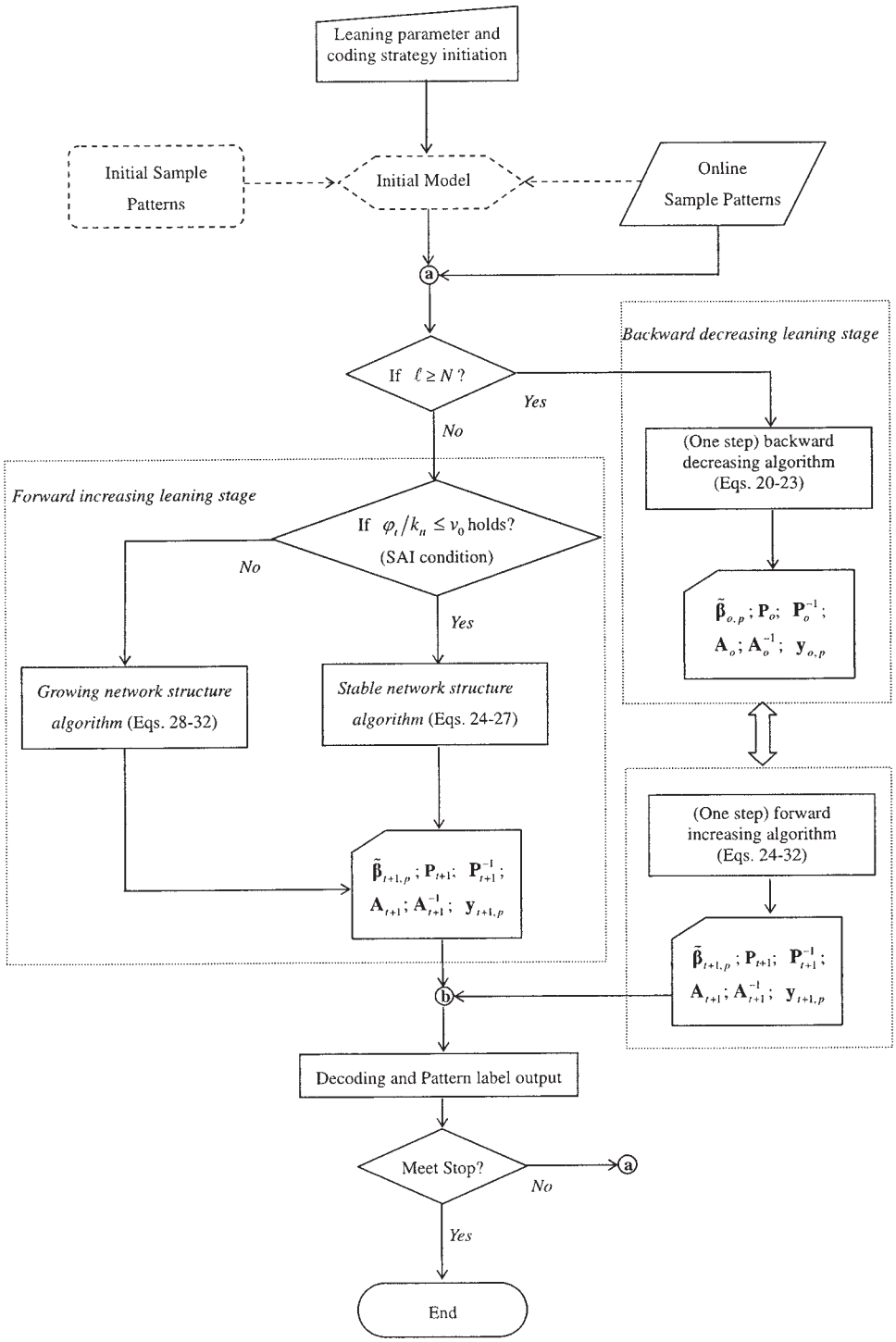
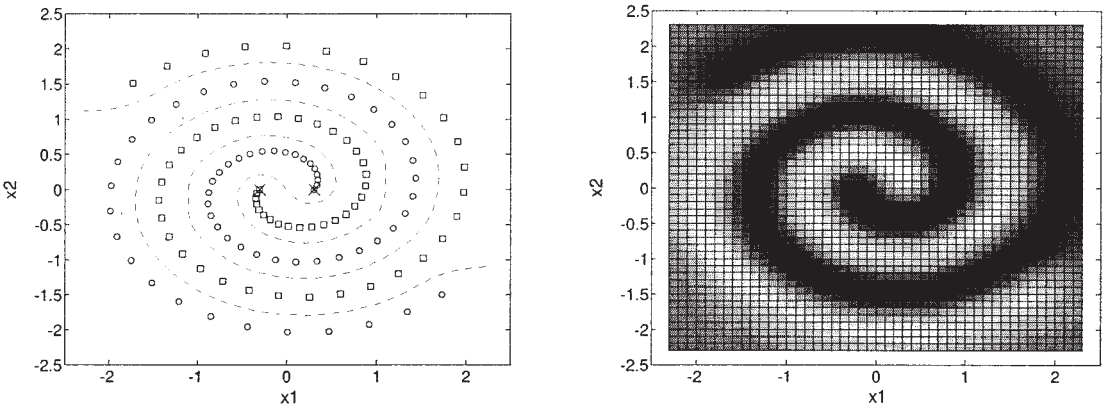


Figure 3. Learning procedure of AKL network for multiclass fault diagnosis.

training data. Mathematically, this scenario indicates the relation of $\sin(\theta_t) \leq \sqrt{v_0}$ holds, although in practice it is verified by calculating $\varphi_t / k_{tt} < v_0$ as in Eq. 13. Because there is no new node added to the network, the structure will be the same, that is, $\tilde{\mathbf{K}}_{t+1} = \tilde{\mathbf{K}}_t$, with only the coefficient vector $\tilde{\boldsymbol{\beta}}_{t,p}$ adjusted online in an optimal sense of Eq. 18. Recursive calculation of Eq. 19 is expressed as

$$\mathbf{A}_{t+1} = [\mathbf{A}_o^T, \mathbf{a}_{t+1}]^T \qquad \mathbf{y}_{t+1,p}^T = [\mathbf{y}_{o,p}^T, y_{t+1,p}]^T \tag{24}$$

where $\mathbf{A}_o \in \mathbb{R}^{(N-1)\times n_t}$ and $\mathbf{y}_{o,p} \in \mathbb{R}^{(N-1)\times 1}$ are defined identically as in Eq. 20 when $\ell \geq N$; otherwise, the first elements of both $\mathbf{A}_o$ and $\mathbf{y}_{o,p}$ should begin from the first training data vector and thus have their corresponding dimensions in this case (that is, when $\ell < N$).



(a) classification boundary (b) global output of the obtained classifier

Figure 4. AKL network classifier for two very similar classes ($\ell = 120$).

To be consistent with the backward learning algorithm, the situation of $\ell \geq N$ is taken as the default in the following derivations. For the case of $\ell < N$ the derivation is identical except for the dimensions of the concerned terms.

By denoting the related quantities, respectively, as $\mathbf{P}_{t+1} = \mathbf{A}_{t+1}^T \mathbf{A}_{t+1}$ and $\mathbf{A}^+ = \mathbf{a}_{t+1}\mathbf{a}_{t+1}^T$, a recursive relation between matrices $\mathbf{P}_o$ (refer to Eq. 21) and $\mathbf{P}_t$ can be obtained as

$$\mathbf{P}_t = \mathbf{P}_o + \mathbf{A}^+ \tag{25}$$

The following derivations are similar to those of backward learning. Again, by applying the Sherman–Woodbury–Morrison formula[40] to Eq. 25, we obtain the inverse relation between matrices $\mathbf{P}_o$ and $\mathbf{P}_{t+1}$ as

$$\mathbf{P}_{t+1}^{-1} = \mathbf{P}_o^{-1} - r^+ \cdot \mathbf{P}_o^{-1}\mathbf{A}^+\mathbf{P}_o^{-1} \tag{26}$$

where $r^+ = 1/(1 + \mathbf{a}_{t+1}^T \mathbf{P}_o \mathbf{a}_{t+1}) \in \mathbb{R}^+$ is a scalar.

By substituting the expressions of Eqs. 24 and 26 into Eq. 19, the forwarding weight vector $\tilde{\boldsymbol{\beta}}_{t,p}$ of the simplified AKL network in a recursive form of $\tilde{\boldsymbol{\beta}}_{o,p}$ is obtained, with an omission of some similar temporal derivations as in Eq. 23

$$\begin{aligned}\tilde{\boldsymbol{\beta}}_{t+1,p} &= \tilde{\mathbf{K}}_{t+1}^{-1}\mathbf{P}_{t+1}^{-1}\mathbf{A}_{t+1}^T\mathbf{y}_{t+1,p} = \tilde{\mathbf{K}}_t^{-1}[\mathbf{P}_o^{-1} - r^+ \cdot \mathbf{P}_o^{-1}\mathbf{A}^+\mathbf{P}_o^{-1}] \\ &\times [\mathbf{A}_o^T\mathbf{y}_{o,p} + a_{t+1}y_{t+1}] = \tilde{\boldsymbol{\beta}}_{o,p} + \tilde{\mathbf{K}}_t^{-1}(\mathbf{P}_{t+1}^{-1}a_{t+1}y_{t+1} \\ &- r^+ \cdot \mathbf{P}_o^{-1}\mathbf{A}^+\tilde{\mathbf{K}}_t\tilde{\boldsymbol{\beta}}_{o,p}) = \tilde{\boldsymbol{\beta}}_{o,p} + \tilde{\mathbf{K}}_t^{-1}\mathbf{s}^+(y_{t+1} - \tilde{\mathbf{k}}_{t+1}\tilde{\boldsymbol{\beta}}_{o,p})\end{aligned} \tag{27}$$

where $\mathbf{s}^+ = \mathbf{P}_{t+1}^{-1}a_{t+1} = \mathbf{r}^+ \cdot \mathbf{P}_o^{-1}a_{t+1}$ is a column vector (its derivation is similar to $\mathbf{s}^-$ in Eq. 23) and $p \in [1, P]$.

Scenario 2: Growing Network Structure. At the early stage of the learning or during process dynamics changes, new nodes will be added automatically to the AKL network. In this case, both the structure and coefficients of the network will be adjusted online. Mathematically, this scenario indicates the relation of $\sin(\theta_t) > \sqrt{v_0}$ holds; in practice it is verified by Eq. 13.

Noting at time $t + 1$ whether the SAI condition of Eq. 11 detects the appearance of a new node (calculated using the only input vector $\mathbf{x}_{t+1}$), the quantities $\tilde{\mathbf{K}}_{t+1}$ and $\mathbf{A}_{t+1}$ will change accordingly. Obviously in this scenario

$$\tilde{\mathbf{K}}_{t+1} = \begin{bmatrix} \tilde{\mathbf{K}}_t & \tilde{\mathbf{k}}_{t+1} \\ \tilde{\mathbf{k}}_{t+1}^T & k_{t+1} \end{bmatrix},$$

with related terms similarly defined in Eq. 11; on the other hand, as the new input vector is inserted as the last node in the AKL network, the following relations hold:

$$\mathbf{a}_{t+1} = [\underbrace{0, \ldots, 0}_{n^t}, 1]^T \in \mathbb{R}^{n_{t+1}\times 1} \qquad \mathbf{A}_{t+1} = \begin{bmatrix} \mathbf{A}_o & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{N\times n_{t+1}} \tag{28}$$

Thus, the following expression can be obtained:

$$\mathbf{A}_{t+1}^T\mathbf{y}_{t+1,p} = \begin{bmatrix} \mathbf{A}_o^T & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{y}_{o,p} \\ y_{t+1,p} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_o^T\mathbf{y}_{o,p} \\ y_{t+1,p} \end{bmatrix} \tag{29}$$

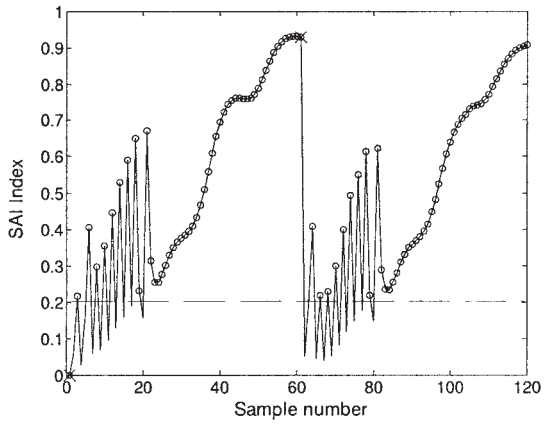
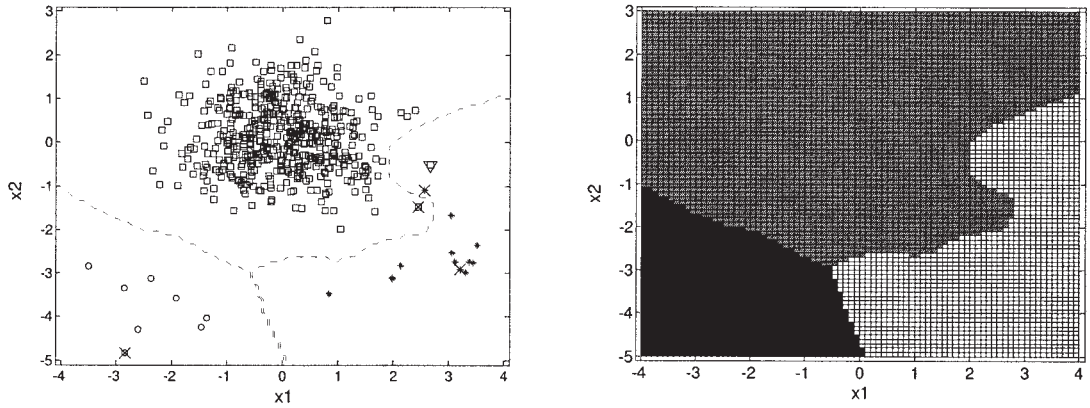


Figure 5. SAI of AKL network classifier for two very similar classes ($\ell = 120$).



(a) classification boundary (b) global output of the obtained classifier

Figure 6. AKL network classifier for three unbalanced classes (ℓ = 541).

Based on the observation of Eq. 28, the matrix $\mathbf{P}_{t+1}$ (and its inverse) in this case can also be constructed in a recursive form, with respect to $\mathbf{P}_o$ (and $\mathbf{P}_o^{-1}$)

$$\mathbf{P}_{t+1} = \begin{bmatrix} \mathbf{P}_o & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \qquad \mathbf{P}_{t+1}^{-1} = \begin{bmatrix} \mathbf{P}_o^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{30}$$

Further, by using the inverse formula for the block matrix,[40] the inverses of $\tilde{\mathbf{K}}_{t+1}$ can be written in a recursive form, with respect to $\tilde{\mathbf{K}}_t$:

$$\tilde{\mathbf{K}}_{t+1}^{-1} = \begin{bmatrix} \tilde{\mathbf{K}}_t & \tilde{\mathbf{k}}_{t+1} \\ \tilde{\mathbf{k}}_{t+1}^T & k_{t+1} \end{bmatrix}^{-1} = \frac{1}{\tilde{\varphi}_{t+1}} \begin{bmatrix} \tilde{\varphi}_{t+1}\tilde{\mathbf{K}}_t^{-1} + \tilde{\mathbf{a}}_{t+1}\tilde{\mathbf{a}}_{t+1}^T & -\tilde{\mathbf{a}}_{t+1} \\ -\tilde{\mathbf{a}}_{t+1}^T & 1 \end{bmatrix} \tag{31}$$

where we denote the SAI combination coefficient vector in Eqs. 11 and 12 as $\tilde{\mathbf{a}}_{t+1} = \tilde{\mathbf{K}}_t^{-1}\tilde{\mathbf{k}}_{t+1}$, to differentiate it from the newly updated one in Eq. 28; its corresponding sine value of SAI angle is $\tilde{\varphi}_{t+1} = k_{t+1,t+1} - \tilde{\mathbf{a}}_{t+1}^T\tilde{\mathbf{k}}_{t+1}$.

The final recursive version of weighting vector $\tilde{\boldsymbol{\beta}}_{t+1,p}$ with respect to $\tilde{\boldsymbol{\beta}}_{o,p}$, is derived by using Eqs. 19 and 29–31:

$$\begin{aligned} \tilde{\boldsymbol{\beta}}_{t+1,p} &= \tilde{\mathbf{K}}_{t+1}^{-1}\mathbf{P}_{t+1}^{-1}\mathbf{A}_{t+1}^T\mathbf{y}_{t+1,p} = \tilde{\mathbf{K}}_{t+1}^{-1}\begin{bmatrix} \mathbf{P}_o^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}\begin{bmatrix} \mathbf{A}_o^T\mathbf{y}_{o,p} \\ y_{t+1,p} \end{bmatrix} \\ &= \frac{1}{\tilde{\varphi}_{t+1}}\begin{bmatrix} \tilde{\varphi}_{t+1}\tilde{\mathbf{K}}_t^{-1} + \tilde{\mathbf{a}}_{t+1}\tilde{\mathbf{a}}_{t+1}^T & -\tilde{\mathbf{a}}_{t+1} \\ -\tilde{\mathbf{a}}_{t+1}^T & 1 \end{bmatrix}\begin{bmatrix} \mathbf{P}_o^{-1}\mathbf{A}_o^T\mathbf{y}_{o,p} \\ y_{t+1,p} \end{bmatrix} \\ &= \frac{1}{\tilde{\varphi}_{t+1}}\begin{bmatrix} (\tilde{\varphi}_{t+1}\mathbf{I} + \tilde{\mathbf{a}}_{t+1}\tilde{\mathbf{k}}_{t+1}^T)\tilde{\boldsymbol{\beta}}_{o,p} - \tilde{\mathbf{a}}_{t+1}y_{t+1,p} \\ y_{t+1,p} - \tilde{\mathbf{k}}_{t+1}\tilde{\boldsymbol{\beta}}_{o,p} \end{bmatrix} \end{aligned} \tag{32}$$

where $p \in [1, P]$.

The forward increasing learning, a one-step-ahead forward learning if $\ell \geq N$, can be considered and obtained as a special case of the basic AKL network algorithm by setting the regularization parameter λ to 0.[30] For the single output case, Engel et al.[32] also proposed a similar strategy to the time series modeling issue. The simplified AKL network proposed here is of two stages and used for the multiclass fault diagnosis issue, equipped with the feature of fixed memory length.

Because all concerned terms in the backward/forward learning stages are derived in recursive form, the weighting vector $\tilde{\boldsymbol{\beta}}_{t+1,p}$ in both Eqs. 27 and 32 can be calculated with a trivial

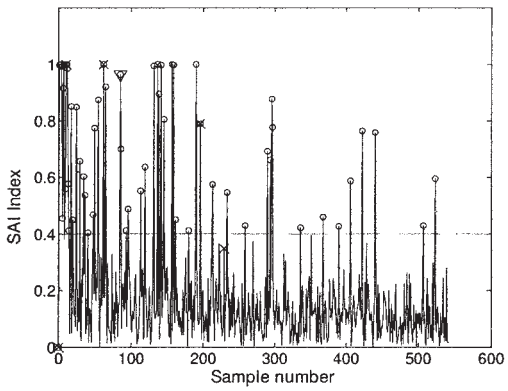


Figure 7. SAI of AKL network classifier for three unbalanced classes (ℓ = 541).

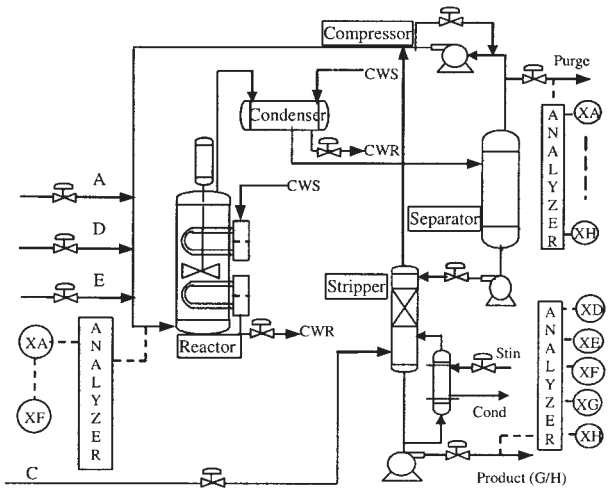


Figure 8. Schema of Tennessee Eastman process.

Table 1. Faults/Classes Considered in the Simulations

Class ID	Fault ID	Description	Type	Fault Duration (h)			
					Testing		
				Learning	Case A	Case B	Case C
C1	Normal	Nominal operating conditions	—	—	—	—	—
C2	IDV1	A/C feed ratio, B composition constant (stream 4)	Step	62–96	78–96	126–144	222–240
C3	IDV2	B composition, A/C ratio constant (stream 4)	Step	110–144	126–144	222–240	270–288
C4	IDV6	A feed loss (stream 1)	Step	158–192	174–192	270–288	126–144
C5	IDV13	Reaction kinetics	Slow drift	206–240	222–240	174–192	78–96
C6	IDV14	Reactor cooling water valve	Sticking	254–288	270–288	78–96	174–192

computation scale. Furthermore, the vector $\tilde{\boldsymbol{\beta}}_{t+1,p}$ has a dimension of the number of nodes in the AKL network, independent of the overall scale of the entire multiclass system, as well as the pattern/data length. This can be an attractive feature for industrial application.

The algorithmic flow sheet of the proposed AKL network is illustrated in Figure 3, involving a backward decreasing algorithm and a forward increasing learning algorithm as stated above. At the beginning stage of online learning (that is, $\ell < N$), only forward increasing learning algorithm will run; when the learning procedure enters into a relatively stable stage ($\ell \geq N$), the backward decreasing and forward increasing learning algorithms will run alternately, each for only one step in a recursive manner.

Before moving to the simulation study, some general guidelines for parameter selection are presented here. There are altogether three parameters for the proposed AKL network, that is, the kernel parameter (such as kernel width σ for Gaussian kernel), the SAI threshold parameter v_0, and the memory control length N. Note that both the selection of input variables and the choice of kernel function type belong to a methodology category, while not a specific parameter of the algorithm. Compared with the other artificial intelligence learning methods, such as neural networks and fuzzy system, the number of parameters of the AKL network is much smaller and thus its trial-and-error selection is possible.

Some model selection (that is, for parameter selection) approaches for traditional SVM or LS-SVM, including the cross-validation and Bayesian inference,[18,19] can provide some insights but are not directly applicable because the AKL network is an online learning algorithm. Rigorous mathematical analysis for this online case is beyond the scope of this article. Fortunately, from an abundance of simulations one can ascertain that the feasible ranges of these three parameters (in a sense that makes the learning error, that is, the minimum of Eq. 18, bounded in an acceptable range) are not so tight and can be easily determined by simulation.

Specifically, the memory length N should be set first. Because the storage space is not a big problem for the proposed algorithm, this factor can be omitted when specifying this parameter. Generally, the memory length should cover at least one period of the process dynamics and be larger than the number of final nodes. Detailed experimental analysis on this point will be presented in the next section; then the threshold v_0 is set (generally lies in a range of [0.05, 0.4]) and the kernel parameter are followed, both determined by simulation.

Table 2. Selected Input Variables to AKL Diagnosis System

Input ID	Variable ID	Variable Description
x1	MEAS1	A feed (stream 1)
x2	MEAS7	Reactor pressure
x3	MEAS9	Reactor temperature
x4	MEAS11	Separator temperature
x5	MEAS18	Stripper temperature
x6	MEAS21	Reactor cooling water outlet temperature

Illustrations

Two numerical cases and the TE benchmark process are studied to illustrate the effectiveness of the proposed AKL network for online classification. All simulations herein are performed with such a software/computer environment: Matlab v6.5 and Pentium 4 CPU of 1.6 GHz with 512M memory. The Gaussian kernel function[18-20] $k(x_1, x_2) = \exp[-\|x_1 - x2\|/(2\sigma^2)]$ is used. With this translation-invariant kernel, the proposed SAI condition reduces to $\varphi_t \leq v_0$, given that $k_{tt} = 1$.

Two numerical examples

Two challenging numerical cases are studied in the two-dimensional space so as to intuitively visualize the performance of the AKL classifier.

Case 1: Classification of Very Similar Patterns. The first diagnosis experiment is to classify two very similar classes,

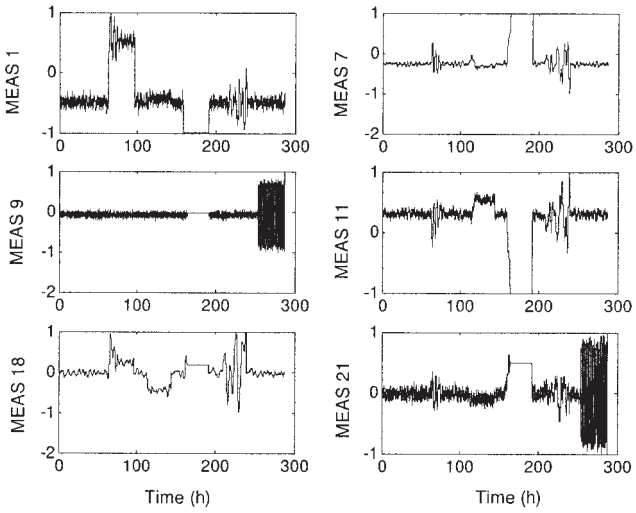
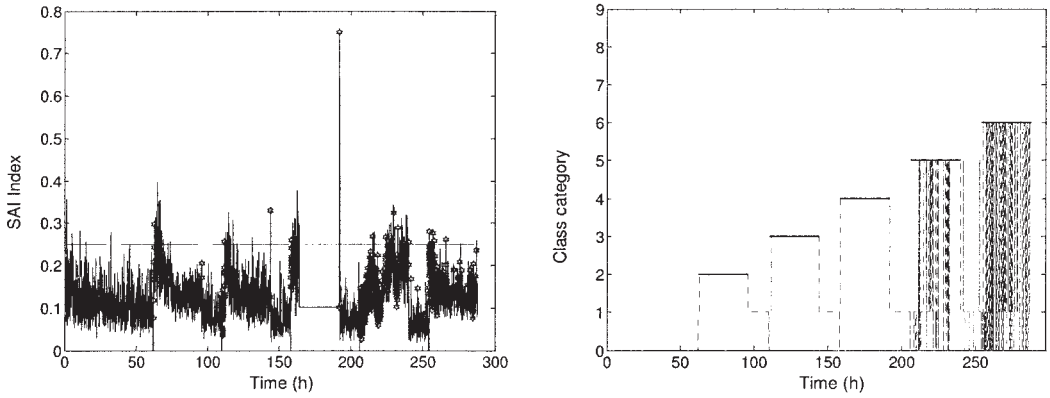


Figure 9. Six basic variables used for fault diagnosis (normalized).



(a) SAI index (b) Online class label prediction

Figure 10. Classification result for TE process based on AKL network (no variable selection and no memory control).

and this is exemplified by using two spiral series. One of them (indicted as a circle) can be regarded as the nominal pattern and the other as a fault pattern (indicted as a rectangle), as shown in Figure 4a. In this experiment, each class has 60 patterns/samples. The spiral series is a benchmark problem in the machine learning field. Although the two series do not overlap, only a strong nonlinear classifier can separate them. This problem is known to be difficult for a traditional multilayer perceptron classifier.[19]

Different from the traditional SVM classification problem, where the training patterns are fed to the learning algorithm in a batch manner and thus the *feeding order* of the patterns is not relevant, the feeding of samples to online machine learning is in a sequential order, and thus the feeding order might affect diagnosis performance. That is, the AKL network "sees" (and then classifies) these 120 patterns in Figure 4a *one by one*, whereas the standard SVM classifiers need to "see" the whole picture at the beginning so as to classify them.

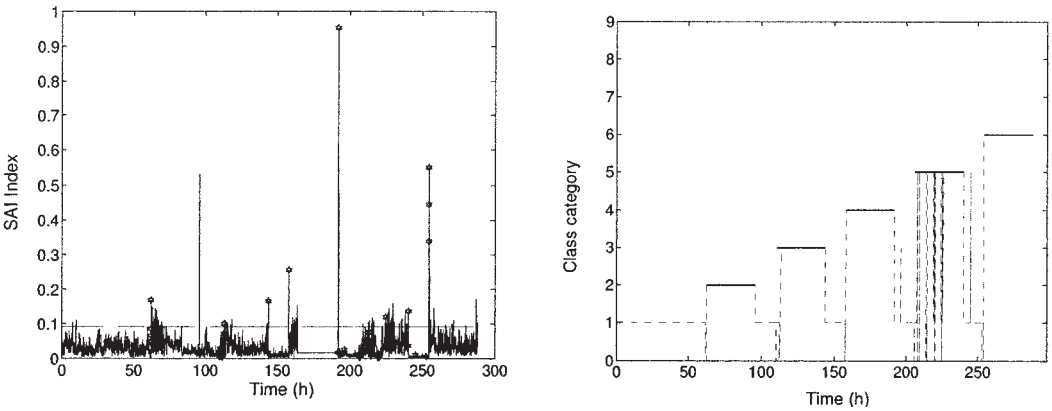
The parameters of AKL are chosen as $\sigma = 0.28$, $v_0 = 0.2$, $\ell = 120$, $N > 120$ (no memory control for this small sample set). These parameter values are simply a pair of feasible ones and by no means optimal. Twenty Monte Carlo simulation experiments are performed, with stochastically generated feeding orders for the 120 patterns to the AKL network.

The classification takes 0.581 s (CPU time in average, including the program initialization). In each experiment, always *only two* samples are misclassified online, that is, the first sample of feeding and the first sample of the other class different from the former. That is to say, the proposed AKL network can correctly classify all samples from the second sample of a class onward, with increasing online learning ability.

Figure 4a shows one of the 20 simulation results, where the 60 nominal patterns (circle) are fed first, and then followed by the other 60 patterns from the fault class (rectangle). The final obtained classification boundary (dashed line) correctly separates these two very similar classes with the largest possible margin. The online misclassified samples are labeled by the cross symbol. The absence of fault samples during the first half of the learning stage does not cause any problem.

The testing (that is, unsupervised learning) result of the obtained classifier is shown in Figure 4b, where the global output in a concerned region is predicted using the obtained classifier. Interestingly, for such a binary classification problem, no decoding is needed for the output of the AKL network and, consequently, some transitional areas exist besides the absolute black and white areas as demonstrated in Figure 4b.

The SAI curve of the above experiment is shown in Figure



(a) SAI index (b) Online class label prediction

Figure 11. Classification result for TE process based on AKL network with variable selection (no memory control).

5 (with the dashed line as its threshold, the same in all the following SAI figures). Altogether, 100 nodes (SVs) are found in this case by the AKL network, which are labeled as circles. Further, the two misclassified patterns are also shown in this figure as cross symbols, where their locations are clearly indicated. In this particular case, the nodes grow until the end and the sparsity of SVs is 83.3%.

Case 2: Classification of Multiclass Unbalanced Patterns. It is well known that the unbalanced patterns (such as the number of patterns belonging to some class is much larger than that belonging to another class) will cause serious classification errors.[18,19] This problem is of importance for fault diagnosis because this situation is very common for industrial process, given that the nominal patterns are always dominant among all the available patterns.

Here a three-class problem is explored and the numbers of these three classes considered are 8, 520, and 13, respectively. Obviously, the patterns from Class 2 are absolutely dominant in the whole pattern set. Figure 6a shows the relative locations of these patterns, where Class 1 is symbolized as a circle, Class 2 as a rectangle, and Class 3 as a star.

Again, 20 Monte Carlo simulation experiments are performed, with *stochastically generated* feeding orders of these unbalanced patterns to the AKL network. The parameters in this case are set as: $\sigma = 0.48$, $v_0 = 0.4$, $\ell = 541$, $N > 541$ (again, no memory control), which are also simply selected by simulation; actually there is obviously no difference compared with those from the neighboring area of these parameters.

The classification takes 3.065 s. Figure 6a shows the classification boundary (dashed line) obtained in one of these experiments, where the classifier indeed gives out a strong nonlinear boundary. One can easily observe that this is also a large margin classifier with excellent generalization ability.[17-20]

There are only five samples misclassified online, indicated as crosses. The average online misclassification number is 4.5 at a ratio of 0.83%. The final classification boundary misclassifies *only one* sample as indicated by the down triangle (interestingly, this sample belonging to Class 2 is correctly classified online). The five online misclassified samples are all correctly categorized by the final boundary. This is explained by the fact that both the classifier itself and its boundary are updated online.

The global test is shown in Figure 6b, where it can be seen, compared with Figure 4b, that the boundaries between different class areas now become more "crisp," that is, without the transitional areas. For industrial process diagnosis, a fault will begin with a transient period before it manifests a definite faulty symptom; thus this kind of *crisp boundary* might have difficulties for early detection and cause misclassification during the process transition. This point will be revisited in the later TE case study.

The SAI curve of the above experiment is shown as Figure 7. Altogether 56 nodes (SVs, the sparsity ratio is 10.37%) are found in this case by the AKL network, labeled as circles, together with the five online misclassified patterns (cross), one final misclassified pattern (down triangle), and their feeding locations.

Tennessee Eastman process

To explore the performance of the AKL classifier under an industrial environment, the Tennessee Eastman (TE) process is used as the experiment platform. The TE benchmark model[42] is based on an actual chemical process and consists of five major units: reactor, condenser, compressor, separator, and stripper; and contains eight components: A, B, C, D, E, F, G, and H. The flow sheet of the TE process is shown in Figure 8. In all simulations, the process sampling time is taken as 3 min for all variables. Details about these variables and defined faults can be found in Downs and Vogel.[42]

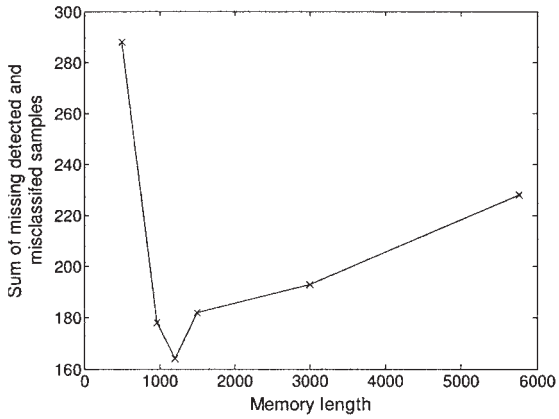


Figure 12. Relationship between memory length and number of mislabeled patterns.

Recall that the objective here is to verify and exemplify the AKL network for process diagnosis, while not to an intensive case study, especially for the TE process (for example, refer to Chiang et al.[3] and Lee et al.[43] for such an aim); altogether five representative faults with different types and locations are considered and tabulated in Table 1. Among them the reaction kinetics variation fault (that is, IDV 13) is more difficult to be diagnosed than other faults because it is a slow draft. This observation is also verified by results of Chiang et al. and Lee et al.[3,43] Thus, special attention is paid to this fault in this case study. Including the nominal pattern, the total number of the considered classes is six. The *unknown pattern* is defined as Class 0, that is, output codes of the AKL network are all "-1"s.

It has been shown in the previous cases that, for the AKL network, both the order and the number of patterns from a class have no obvious effects on the classification results. Therefore, without loss of generality, all the simulated faults for the *learning stage* (supervised) occur at the 14th hour of each separate 48-h operation run and last to the end to simplify the programming (that is, $6 \times 48 = 288$ h for six cascade runs; their specific duration time is listed in Table 1, "learning" column).

Scenario 1: Effect from Input Variable Strategy. Obviously the selection of input variables to the classifier has a significant effect on its classification performance.[3] In this article, the input variable to the AKL network is selected according to the process knowledge and operation characteristics. This is feasible for a chemical process because such information is usually available and many successful applications have been reported.[3,24-26]

Lee et al.[43] applied the signed digraph and PLS to the TE process and obtained the cause–effect relation between different faults and measurement variables. Their so-called *target variables*

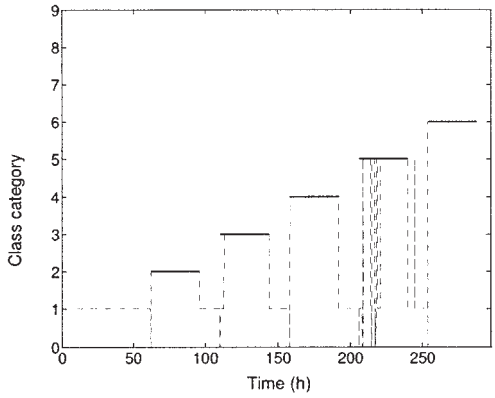


Figure 13. Online class label prediction with variable selection and memory control (N = 1200).

provide the process information needed here (refer to their Tables 4 and 6 for details) and altogether eight measured variables are concerned with the fault set considered in Table 1.[43]

However, only six variables among them are adopted as the basic input variables to the AKL classifier. These variables are tabulated in Table 2 and Figure 9 shows their measurements (normalized) during a learning period of six cascade operation runs. The other two excluded variables are MEAS 8 and 13, corresponding to the reactor level and separator pressure, respectively. The deletion of MEAS 8 is the result of its slow dynamic response, and for MEAS 13 it is to alleviate the potential pattern overlaps because IDV 13 is already well explained by several other variables in Table 2.

From Figure 9 one can confirm that the faulty information is indeed reflected at different operational stages. For example, the fault of IVD 6 (the loss of a feed) rapidly drives the process to violate the operational limits and consequently shut down; however, it should be noted that the occurrence of IDV 13 is at the 206th hour (see Table 1, "learning" column); it is until about 1–3 h later that the three affected variables (MEASs 7, 11, and 18) obviously begin to fluctuate. Because one does not have such delay information a priori, in the supervised learning procedure a class label is simply allocated to the corresponding online pattern 15 min after the occurrence of the corresponding fault. This may cause some artificial misclassifications, as illustrated in the following simulations.

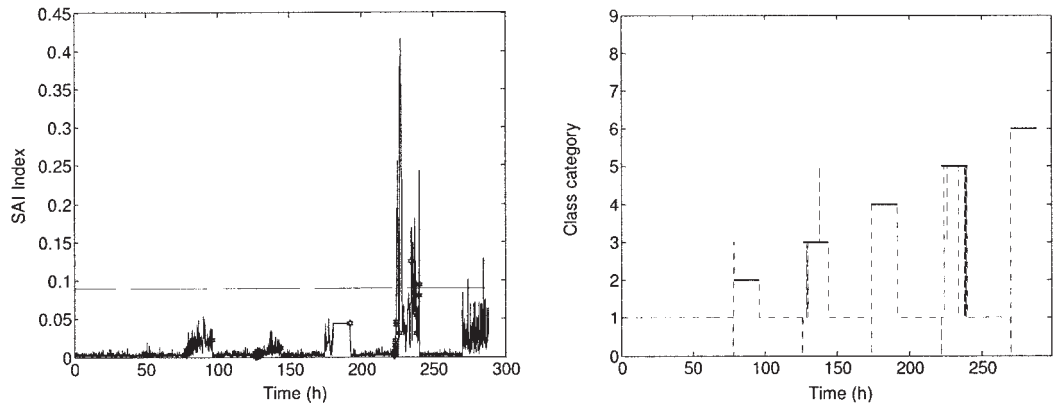
On the other hand, to incorporate the autocorrelation of variables into the classifier to improve its performance, Chiang et al.[26] recommended that, in addition to the variable selection, the one- to two-step(s) time lag of these selected variables should also be added to constitute the final generalized input variables. Here, the one-step time-lag strategy is adopted for the AKL network learning to constitute the generalized input $\mathbf{x_t}$ of Eq. 7, such as $\mathbf{x_t} \in \mathbb{R}^{12}$.

For comparison, classification for the TE process *without variable selection and no memory control* is first explored and the results are shown in Figure 10. All of the 22 continuous process variables[42] (as well as their one-step time-lag quantities) are used as input patterns. A satisfying parameter set for the AKL network used in this case includes $\sigma = 2.20$, $v_0 = 0.25$, $\ell = 5760$ (that is, 288 h), $N > 5760$ (that is, no memory control).

The AKL network takes 191.145 s to classify this 288-h operation run. Altogether 146 nodes are found and the sparsity ratio is 2.54%. Because the classification results cannot now be directly visualized, both the missed detection and misclassified patterns (see below for definition) are labeled out as hexagrams on the SAI index curve in Figure 10a; the plot of online class label prediction (dashed line) and ideal label (the short solid line) are compared in Figure 10b, giving the real-time classification performance.

Figure 10a shows there are many mislabeled patterns, particularly in the last two stages when IDVs 13 and 14 occur. The number of misclassification patterns amounts to 166 (excluding those categorized as Class 1), with a global misclassification rate of 0.049; the number of missed detection patterns is 359, with a global missed detection rate of 0.106. This poor performance is explained by the fact that too many unrelated variables are included, thus degrading the diagnosis performance. Here the *misclassification rate* is defined as $\eta_{mc} = N_{mc}/M$, where N_{mc} is the number of misclassified patterns and M is the duration of a fault; the *missed detection rate* is defined as $\eta_{md} = N_{md}/M$, where N_{md} is the number of patterns that are classified as Class 1 during occurrence of a fault.

The effect from *variable selection* alone to the classification



(a) SAI index (b) Online evolution of class label prediction

Figure 14. Unsupervised Class label prediction for TE process using AKL network diagnosing model (Case A).

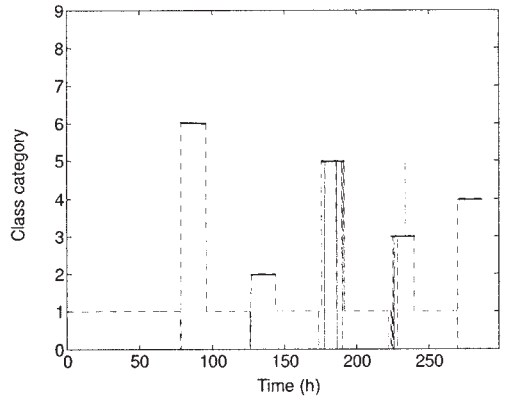


Figure 15. Unsupervised Class label prediction for TE process using AKL network diagnosing model (Case B).

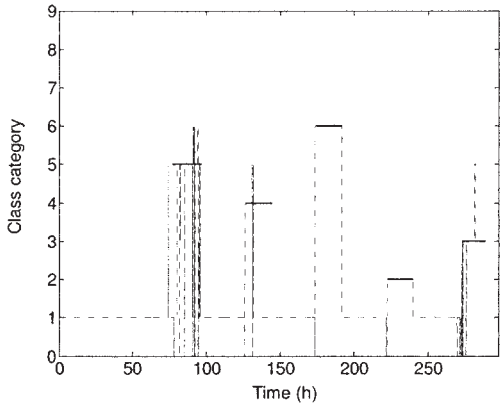


Figure 16. Unsupervised Class label prediction for TE process using AKL network diagnosing model (Case C).

performance is then explored. Figure 11 shows the online classifying result with six selected input variables in Table 2 (although no memory length control is applied in this case, that is, $N > \ell$). A satisfying parameter set for the AKL network in this case includes $\sigma = 0.80$ and $v_0 = 0.090$.

The AKL network takes 136.546 s to classify this 288-h operation run. Both the missed detection and misclassification patterns are labeled as hexagrams on the SAI index curve in Figure 11a. Altogether 121 nodes are found and the sparsity ratio is about 2.10%. Meanwhile its class label prediction plot is presented in Figure 11b to show the real-time classification performances. From Figure 11a one can see the diagnosing performance is greatly improved, in comparison with Figure 10a, especially for the last two stages where IDVs 13 and 14 occur. In fact, the total number of misclassifications in this case is 52, that is, a global misclassification rate of 0.015; the missed detected patterns amount to 176, yielding a global rate of 0.052. This significant improvement indicates the importance of variable selection for pattern learning methods.

On the other hand, both Figures 10b and 11b indicate an interesting result that, for the AKL classifier, most of the misclassified patterns are categorized as unknown patterns (Class 0), although not as other fault categories.

Scenario 2: Effect from Memory Length. Intuitively, the effect of memory length may not be so notable, compared with that of variable selection. Nevertheless, it is still important to the diagnosis performance of the AKL network. Rigorous mathematical analysis of its influence on the classification performance is beyond the scope of this article. An empirical guideline on the determination of memory length has been stated at the end of Section 3 and here the focus is on the experimental analysis.

Simulation experiments are performed with different memory (control) lengths, with the same AKL network parameters and input used in the latter case in Scenario 1 (that is, in Figure 11). Figure 12 presents the variation trend between the sum of missed detection and misclassification patterns and the memory length. This comparative experiment shows that an optimal memory length indeed exists and in this case the value is around 1200. Furthermore, it is also shown that using too short a memory length is no better than no memory control. In fact, if a value of $N < 400$ (that is, 20 h) is used, the misclassification ratio will drastically increase to an unacceptable degree.

Next, the AKL network with both *variable selection* (that is, Table 2) and "*optimal*" *memory length* (that is, $N = 1200$) is used. The online diagnosing result is shown in Figure 13. The AKL network takes 175.005 s to classify this 288-h operation run.

The SAI curve is identical to Figure 11a and thus not shown here again (of course, the numbers and positions of missed detection and misclassification patterns are different in these two cases). The total number of misclassifications in this case is 39, that is, a global misclassification rate of 0.011; the missed detected patterns amounts to 125, yielding a global rate of 0.037. This manifests the achievement of a further improvement, compared with the result of Figure 11.

Again it can be found in Figure 13 that the missed detection/misclassification occurs usually at the beginning period of a fault. Just as previously mentioned, during this kind of transient period the symptom of a fault is still not obvious. Thus, too early labeling of the patterns during these transient periods will unavoidably lead to missed detection or misclassification. This phenomenon is particularly obvious for faults with a large transient period, such as the slow shift fault of reaction kinetics (IDV 13).

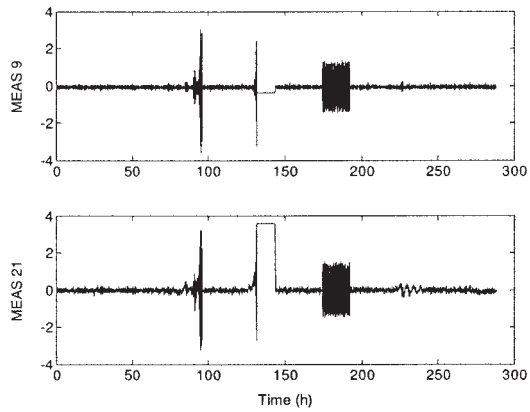


Figure 17. Dynamic response of MEASs 9 and 21 (normalized, testing stage of Case C).

Table 3. Detection Rapidity of AKL Classifier

	Detection Delay (min)			
	AKL Network			
Fault ID	Learning	Testing	Lee et al.	Chiang et al.
IDV1	15/30	27/33	36/36	6/21
IDV2	93/180	201/219	78/115	36/75
IDV6	3/30	3/3	1/1	3/33
IDV13	120/228	108/111	130/142	111/147
IDV14	9/405	3/6	11/11	3/18

Scenario 3: Unsupervised Learning Performance. To verify the generalization ability of the diagnosing model obtained in Scenario 2 (as Figure 13), it is applied to predict the class label information for new operation runs *without online supervised labeling and learning*. The testing experiments include three cases and, without loss of generality, each simulated fault for testing experiments occurs at the 30th hour of each corresponding run to simplify the programming (refer to the "testing" column in Table 1 for the time duration of each fault).

Case A: Using the Same Rand Noise Seeds for Testing as That of Chiang et al.[3] Because the property of a fault (such as magnitude and dynamic behavior) cannot be directly modified in the routine of Downs and Vogel,[42] a simple (although perhaps not so sufficient) way to generate the testing data is changing the rand noise seed of each fault simulation, of course, as well as its occurrence time and duration (listed in Table 1). Here the same rand noise seeds for testing as that of Chiang et al.[3] are used (available at http://brahms.scs.uiuc.edu).

The AKL network takes 54.368 s to predict the class labels for this 288-h operation run, and both the missed detection and misclassified patterns are labeled as hexagrams on the SAI index curve in Figure 14a. An online evolution plot for class label prediction is provided in Figure 14b, showing the *unsupervised* classification performance. The total number of misclassifications in this case is 38, that is, a global misclassification rate of 0.021; the missed detected patterns amounts to 87, yielding a global rate of 0.048.

Case B: Changing the Set Point of the 8th Control Loop. In this testing experiment, besides the setting for Case A, the set point of the 8th control loop, which regulates the level of stripper (that is, MEAS 15) by the stripper liquid product flow (that is, XMV 8),[42] is changed from the original 50 to 48.7 (unit: %). This change results in the dynamic characteristics of the considered fault set changing correspondingly and thus differentiates by some degree to the patterns in learning experiments, explored in the previous Scenarios 1 and 2.

The time duration (or more specifically, the occurring order) of each considered fault is listed in Table 1. The online evolution plot for class label prediction (unsupervised) is presented in Figure 15. The total number of misclassifications in this case is 42, that is, a global misclassification rate of 0.023; the missed detected patterns amounts to 88, yielding a global rate of 0.049. Both this result and that obtained in Case A indicate the developed AKL classifier has excellent generalization ability.

Case C: Changing the Controller Parameters of the 18th Control Loop. Given that both the controlled and manipulated variables of control loop 8 are not directly related to input variables of the AKL classifier, the diversity of fault patterns in Case B might also be insufficient. To explore the worst-case diagnosing performance of the AKL classifier under extensive changes of fault properties, here the controller parameters of the 18th control loop, which regulates the temperature of reactor (that is, MEAS 9) by the reactor cooling water outlet temperature (that is, MEAS 21),[42] are changed from the original 28.3 to 31.0 (for the gain) and from 982 to 960 (for the integral time, unit: s), respectively.

The time duration of each considered fault is listed in Table 1, and the online evolution plot for class label prediction (unsupervised) is presented in Figure 16. The total number of misclassifications in this case is 84, that is, a global misclassification rate of 0.047; the missed detection patterns amount to 356, yielding a global rate of 0.198.

It can be seen in Figure 16 the most missed detections and misclassifications happen to the first two faults, that is, the IDV 13 and IDV 6. By studying the dynamic responses of MEAS 9 and MEAS 21 shown in Figure 17, one can verify that the TE process becomes unstable when these two faults occur, which causes the input patterns to the AKL classifier to change dramatically. This is a commonly recognized challenge for pattern analysis based methods, whose performances essentially depend on the degree of variation of the input patterns relative to the training patterns. Fortunately, in this case the patterns of other faults, especially the IDVs 1 and 5, are not significant changed under the new control parameters and thus the AKL classifier still successfully diagnosed them, as shown in Figure 16.

Tables 3–5 present a broad overview on the diagnosis performance of the proposed AKL network classifier. Performance indices from the other two fault diagnosing strategies applied to the TE process [that is, the signed digraph method combined with PLS modeling[43] and the multiple multivariate SPM and Fischer discriminant analysis (FDA) based methods[3]] are also listed in the corresponding tables. However, it should be noted, because the methodologies, technical parameters, and

Table 4. Missed Detection Performances of AKL Classifier

	Missed Detection Rate						
	AKL Network						
	Learning Experiment*			Testing Experiment			
Fault ID	Scenario 1a	Scenario 1b	Scenario 2	Case A	Case B	Case C	Chiang et al.
IDV1	0.013	0.006	0.006	0.008	0.008	0.008	0.000/0.008
IDV2	0.046	0.087	0.084	0.125	0.125	0.122	0.010/0.026
IDV6	0.015	0.001	0.001	0.000	0.000	0.694	0.000/0.013
IDV13	0.288	0.162	0.090	0.108	0.111	0.164	0.040/0.060
IDV14	0.166	0.003	0.003	0.000	0.000	0.000	0.000/0.158

*Scenario 1a and Scenario 1b correspond to the case without variable selection (Figure 10) and that with variable selection (Figure 11), respectively.

Table 5. Misclassification Performances of AKL Classifier

	Misclassification Rate						
	AKL Network						
	Learning Experiment			Testing Experiment			
Fault ID	Scenario 1a	Scenario 1b	Scenario 2	Case A	Case B	Case C	Chiang et al.
IDV1	0.000	0.000	0.000	0.019	0.019	0.019	0.013/0.0880
IDV2	0.003	0.001	0.004	0.069	0.075	0.078	0.010/0.441
IDV6	0.003	0.003	0.003	0.003	0.003	0.017	0.000/0.834
IDV13	0.132	0.068	0.040	0.011	0.017	0.117	0.208/1.000
IDV14	0.106	0.004	0.007	0.003	0.003	0.003	0.001/0.998

experiment designs of these three methods are quite different, a rigorous and fair comparison is impracticable, and the results from the other two methods shown here serve mainly for a conceptive aim.

Among them, Table 3 gives the fault detection speeds for these three different diagnosing strategies. The left and right values of the slash in this table represent the best/worst results of the corresponding method. However, the sampling time of Lee et al. is 1 min (for the other two, 3 min) and they did not use the persistent trigger rule (the other two both use a rule of six-point violation).

For the AKL classifier, its values are further specified as learning and testing stages, obtained from Scenarios 1 and 2 and Cases A–C (in Scenario 3), respectively. Obviously, its worst results (including the following Tables 4 and 5) almost all come from Scenario 1 (the case without variable selection) or Case C, which can also be observed from Figures 10 and 16. On the whole, it can be seen that these three methods have a comparable performance on the best detection speed.

Tables 4 and 5 present the missed detection rate and the misclassification rate of the proposed method and that of Chiang et al.,[3] respectively, because both of them use these two performance indices. Nevertheless, the results of Chiang et al.[3] in Tables 4 and 5 were obtained by applying the multiple multivariate SPM/FDA based methods to an experimental setting similar to Case A; on the other hand, Lee et al.[43] proposed to use the indices of *accuracy*, *wrong detection*, and *resolution* for their signed digraph based diagnosing method. Because their method gives out a *fault set*, whereas not a single candidate fault is the diagnosis result, one cannot directly compare it with the other two methods and is thus omitted in Tables 4 and 5.

With respect to the missed detection rate, the testing results of AKL classifier on IDVs 6 and 14 in Table 4 show the same performance as that of the best results obtained from the many methods explored in Chiang et al.,[3] whereas the latter gives a lower missed detection rate for the other three faults. However, as just mentioned previously, there is always a trade-off between the missed detection rate and the misclassification rate. Table 5 indicates the AKL classifier gives a much lower misclassification performance on the IDV 13, compared with the best result of Chiang et al.,[3] and meanwhile comparable performances on both IDVs 1 and 2 are also observed. Nevertheless, it should be emphasized that the AKL classifier is developed by online learning with one-by-one approximating patterns, that is, beginning from a very small and incomplete training set.

On the other hand, in Tables 3–5 some entries in the testing stage are better than their corresponding entries in the learning stage, which is different from the batch training method whose testing performances are generally worse than that of training. In fact, Figures 4b and 6b also show improved performances compared with their online supervised learning counterparts.

Conclusions

This article has demonstrated that process diagnosing by use of a statistical pattern recognition approach is feasible and promising. A unified least-square kernel (ULK) framework has been proposed that can solve both the problems of (MIMO) process modeling and (multiple) faults diagnosis by a single learning algorithm. To meet the practical requirements of an industrial diagnosis system, an adaptive kernel learning (AKL) network as a specific algorithmic implementation of ULK strategy is proposed. The AKL is implemented as a two-stage online learning algorithm with a fixed memory length, in the recursive form of a multiclass kernel learning machine. It is shown that the proposed AKL network performs well with the challenging numerical cases and the TE process. Technical details and application issues have been discussed and highlighted as well.

Acknowledgments

This work was sponsored by the National Natural Science Foundation of China (Project Nos. 20576116 and 20206028) and Alexander von Humboldt Foundation (H. Q. Wang), which are gratefully acknowledged.

Literature Cited

1. Venkatasubramanian V, Rengaswamy R, Yin KW, Kavuri SN. A review of process fault detection and diagnosis: Parts I–III. *Comput Chem Eng.* 2003;27:293-346.
2. Frank PM, Ding SX. Survey of robust residual generation and evaluation methods in observer based fault detection systems. *J Process Control.* 1997;7:403-424.
3. Chiang LH, Russell EL, Braatz RD. *Fault Detection and Diagnosis in Industrial Systems.* London, UK: Springer-Verlag; 2001.
4. Wang XZ. *Data Mining and Knowledge Discovery for Processing Monitoring and Control.* Berlin: Springer-Verlag; 1999.
5. Kresta J, MacGregor JF, Marlin TE. Multivariable statistical monitoring of process operating performance. *Can J Chem Eng.* 1991;69:35-47.
6. Huang Y, Gertler J, McAvoy T. Sensor and actuator fault isolation by structured partial PCA with nonlinear extensions. *J Process Control.* 2000;10:459–469.
7. MacGregor JF, Jaeckle C, Koutoudi MK. Process monitoring and diagnosis by multiblock methods. *AIChE J.* 1994;40:826-839.
8. Wang HQ, Song ZH, Li P. Fault detection behavior and performance analysis of principal component analysis based process monitoring methods. *Ind Eng Chem Res.* 2002;41:2455-2464.
9. Lu N, Gao FR, Yang Y, Wang F. PCA-based modeling and on-line monitoring strategy for uneven-length batch processes. *Ind Eng Chem Res.* 2004;43:3343-3352.

10. Venkatasubramanian V, Chan K. A neural network methodology for process fault diagnosis. *AIChE J.* 1989;35:1993-2002.
11. Akbaryan F, Bishnoi PR. Fault diagnosis of multivariate systems using pattern recognition and multisensor data analysis technique. *Comput Chem Eng.* 2001;25:1313-1339.
12. Dash S, Rengaswamy R, Venkatasubramanian V. Fuzzy-logic based trend classification for fault diagnosis of chemical processes. *Comput Chem Eng.* 2003;27:347-362.
13. Kramer MA, Palowitch BL. A rule-based approach to fault diagnosis using the signed directed graph, *AIChE J.* 1987;33:1067-1078.
14. Aronszajn N. Theory of reproducing kernels. *Trans Am Math Soc.* 1950;68:337-404.
15. Vapnik VN. An overview of statistical learning theory. *IEEE Trans Neural Netw.* 1999;10:988-999.
16. Agrawal M, Jade AM, Jayaraman VK, Kulkarni BD. Support vector machines: A useful tool for process engineering applications. *Chem Eng Prog.* 2003;98:100-105.
17. Vapnik VN. *The Nature of Statistical Learning Theory*. New York: Springer-Verlag; 1995.
18. Schölkopf B, Smola AJ. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA: MIT Press; 2002.
19. Suykens JAK, Van Gestel T, De Brabanter J, De Moor B, Vandewalle J. *Least Squares Support Vector Machines*. Singapore: World Scientific; 2002.
20. Taylor JS, Cristianini N. *Kernel Methods for Pattern Analysis*. Cambridge, UK: Cambridge Univ. Press; 2004.
21. Desai K, Badhe Y, Tambe SS, Kulkarni BD. Soft-sensor development for fed-batch bioreactors using support vector regression. *Biochem Eng J.* 2006;27:225-239.
22. Yan W, Shao H, Wang X. Soft sensing modeling based on support vector machine and Bayesian model selection. *Comput Chem Eng.* 2004;28:1489-1498.
23. Lee DE, Song JH, Song SO, Yoon ES. Weighted support vector machine for quality estimation in the polymerization process. *Ind Eng Chem Res.* 2005;44:2101-2105.
24. Trafalis TB, Oladunni O, Papavassiliou DV. Two-phase flow regime identification with a multiclassification support vector machine (SVM) model. *Ind Eng Chem Res.* 2005;44:4414-4426.
25. Jemwa GT, Aldrich C. Improving process operations using support vector machines and decision trees. *AIChE J.* 2005;51:526-543.
26. Chiang LH, Kotanchek ME, KordonAK. Fault diagnosis based on Fisher discriminant analysis and support vector machines. *Comput Chem Eng.* 2004;28:1389-1401.
27. Fung G, Mangasarian OL. Proximal support vector machine classifiers. In: Provost F, Srikant R, eds. Proceedings of Knowledge Discovery and Data Mining, San Francisco, CA; 2001:77-86.
28. Rifkin R, Yeo G, Poggio T. Regularized least squares classification. In: Suykens JAK, Horvath G, Basu S, Micchelli C, Vandewalle J, eds. *Advances in Learning Theory: Methods, Model and Applications* (NATO Science Series III: Computer and Systems Sciences). Amsterdam: VIOS Press; 2003;Chapter 7:131-154.
29. Allwein EL, Schapire RE, Singer Y. Reducing multiclass to binary: A unifying approach for margin classifiers. *Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco, CA: Morgan Kaufmann; 2000:9-16.
30. Wang HQ, Song ZH, Li P, Ding SX. AKL networks for industrial analyzer modeling and fault detection. Proceedings of the IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (safeprocess2006). Beijing, China; 2006.
31. Crammer K, Singer Y. On the learnability and design of output codes for multiclass problems. *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*. San Francisco, CA: Morgan Kaufmann; 2000:35-46.
32. Engel Y, Mannor S, Meir R. Kernel recursive least squares. Available at http://www.ee.technion.ac.il /~rmeir/ Publications/KrlsReport.pdf; 2003.
33. Csató L, Opper M. Sparse representation for Gaussian process models. In: Leen TK, Dietterich TG, Tresp V, eds. *Advances in Neural Information Processing Systems*. Volume 13. Cambridge, MA: MIT Press; 2001:444-450.
34. Schölkopf B, Knirsch P, Smola AJ, Burges A. Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces. In: Ahler R-J, Levi P, Schanz M, May F, eds. *Proceedings of the 20th DAGM Symposium Mustererkennung 1998*. Berlin: Springer-Verlag; 1998:124-132.
35. Drineas P, Mahoney MW. On the Nyström method for approximating a gram matrix for improved kernel-based learning. *J Machine Learn Res.* 2005;6:2153-2175.
36. Cauwenberghs G, Poggio T. Incremental and decremental support vector machine learning. In: Leen TK, Dietterich TG, Tresp V, eds. *Advances in Neural Information Processing Systems*. Volume 13. Cambridge, MA: MIT Press; 2001;409-415.
37. Shilton A, Palaniswami M, Ralph D, Tsoi AC. Incremental training of support vector machines. *IEEE Trans Neural Netw.* 2005;16:114-131.
38. Li Y, Long PM. The relaxed online maximum margin algorithm. In: Solla SA, Leen TK, Müller K-R, eds. *Advances in Neural Information Processing Systems*. Volume 12. Cambridge, MA: MIT Press; 1999: 498-504.
39. Kivinen J, Smola AJ, Williamson RC. Online learning with kernels. *Advances in Neural Information Processing Systems*. Volume 14. Cambridge, MA: MIT Press; 2002:785-793.
40. Golub GH, Van Loan CF. *Matrix Computations*. 3rd Edition. Baltimore, MD: The John Hopkins Univ. Press; 1996.
41. Franc V, Hlavac V. Training set approximation for kernel methods. Proceedings of the 8th Computer Vision Winter Workshop, Prague, Czech Republic, Czech Pattern Recognition Society; 2003;121-126.
42. Downs JJ, Vogel EF. A plant-wide industrial process control problem. *Comput Chem Eng.* 1993;17:245-255.
43. Lee G, Han CH, Yoon ES. Multiple-fault diagnosis of the Tennessee Eastman process based on system decomposition and dynamic PLS. *Ind Eng Chem Res.* 2004;43:8037-8048.

Manuscript received Apr. 20, 2006, and revision received July 25, 2006.