Diagnostic Pattern Recognition on Gene-Expression Profile Data by Using One-Class Classification

Yun Xu and Richard G. Brereton*

School of Chemistry, University of Bristol, Cantock's Close, BRISTOL BS8 1TS, United Kingdom

Received September 7, 2004

In this paper, we perform diagnostic pattern recognition on a gene-expression profile data set by using one-class classification. Unlike conventional multiclass classifiers, the one-class (OC) classifier is built on one class only. For optimal performance, it accepts samples coming from the class used for training and rejects all samples from other classes. We evaluate six OC classifiers: the Gaussian model, Parzen windows, support vector data description (with two types of kernels: inner product and Gaussian), nearest neighbor data description, *K*-means, and PCA on three gene-expression profile data sets, those being an SRBCT data set, a Colon data set, and a Leukemia data set. Providing there is a good splitting of training and test samples and feature selection, most OC classifiers can produce high quality results. Parzen windows and support vector data description are "over-strict" in most cases, while nearest neighbor data description is "over-loose". Other classifiers are intermediate between these two extremes. The main difficulty for the OC classifier is it is difficult to obtain an optimum decision threshold if there are a limited number of training samples.

1. INTRODUCTION

Gene expression profiling by DNA microarrays is a powerful new technology that can measure thousands of gene expression levels simultaneously. It provides abundant information and is very useful for clinical diagnosis. A number of methods have already been applied in gene-expression profile analysis.[1−3] In this paper, we consider the problem in another way: how can we cope with samples that do not belong to any classes used in the training set? In clinical diagnostic applications, it is possible to test an unknown sample with a certain classifier that actually may be not suitable for it (e.g., test a normal tissue on a cancer classifier). If that classifier does not have a rigorous outlier detection capability, it will definitely produce an incorrect result. For multiclass classifiers, which are most commonly used, it is often difficult to detect a sample that does not belong to any of the classes used in the training stage. Every sample tested with it will be forced into membership of one class (on the basis of the similarity of features), regardless of whether it truly belongs to that class or not. By evaluating the numerical output of the multiclass classifier, it is still possible to identify outliers, but this process is usually not straightforward and is sometimes time-consuming (e.g., in ref 1, the authors build more than 3000 neural network models to estimate the distribution of the numerical output of the sample of interest). Moreover, it is often necessary to extend the current classifier to accommodate more new classes. For multiclass classifiers, one usually needs to rebuild whole class models. We try to overcome this problem in another way, building a classifier on a single class; it accepts most samples coming from that class and rejects samples coming from other classes. This problem is defined as one-class (OC) classification (data description). For this kind of classifier, it is both easy to identify outliers and convenient to extend to more classes.

In this paper, we follow the terminology for one-class classification: the samples we have used to train the classifier are called *target samples*; the class those samples belong to is called the *target class*. Any samples that do not belong to the *target class* are called *outliers*; we can also say that they belong to the *outlier class*. We first overview some one-class classifiers then apply them on three well-known gene-expression profile data sets and compare their performances.

2. THEORY

One-class classification can be considered as a specific extension of conventional outlier detection for pattern recognition purposes. Many outlier detection algorithms as well as one-class classifiers have already been proposed in past decades.[4−9] For OC classifiers, according to how they achieve their objective, they can be classified into three families: density estimation, boundary methods, and reconstruction methods.

Density estimation is the most comprehensive way for data description. Given a sufficient number of samples, one can estimate the distribution of the target class and set a threshold probability for rejection. Generally, a local density estimation such as Parzen windows[6] is more flexible than a global density estimation such as Gaussian models[6] and, therefore, yields better results in most practical applications. However, such methods usually need a large number of training samples (especially for local density estimations) to get a reliable estimation. Also, methods belonging to this family usually have difficulties performing well if there is a high feature space.[6]

The boundary methods achieve the objective in another way. Instead of estimating the distribution of the target class,

* Corresponding author. Tel.: +44-117-0287658. E-mail: r.g.brereton@bris.ac.uk.

10.1021/ci049726v CCC: $30.25
Published on Web 07/28/2005

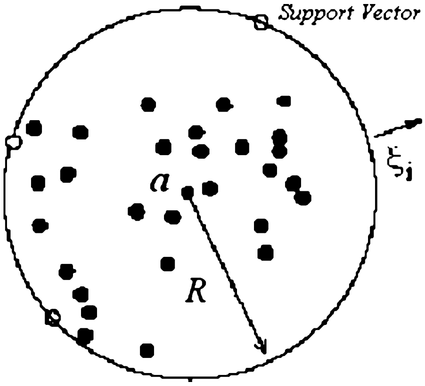

Figure 1. A 2-D example of SVDD. A circle (decision boundary) is drawn around the target samples. This circle is determined by two parameters: center a and radius R. Slack variable ξ controls the fraction of the training samples remaining outside the boundary; in optimization, ξ is controlled by penalty term C.

they draw a boundary around a target class that covers most target samples, and so, fewer samples are needed for training compared to density estimation models. However, these methods usually assume target samples are well-sampled and are representative of the overall class; if this is not the case, the boundary will be biased toward training samples. Typical methods belonging to this family are nearest neighbor data description (NNDD)[7] and support vector data description (SVDD).[8]

For reconstruction methods, one first builds a model to fit a series of training samples then describes a test sample in terms of that model. The construction error can be used as a discriminator for the target and outlier classes. A large number of methods in this family are derived from partitional clustering algorithms (e.g., K-means, learning vector quantization, self-organizing map, etc.). Another type of method belonging to this family is derived from algorithms that try to summarize data points in subspaces such as PCA.

We will briefly describe some one-class methods that are used in this paper below.

2.1. Density Estimation. *2.1.1. Gaussian Model.* This is the simplest method in this family. Assuming all target samples coming from a d-dimensional target class are derived from one prototype and perturbed by numerous small independent sources of noise, they should form a multivariate normal distribution, and the probability density function for observing the data given that a d-dimensional test sample belongs to a predefined class can be calculated from the following equation:

$$p(z{:}\mu, \sum) = \frac{1}{(2\pi)^{d/2}|\sum|^{1/2}}\,\mathrm{e}^{\{-1/2(z-\mu)^T\sum^{-1}(z-\mu)\}}$$

where μ and Σ are the mean and covariance matrix of the target class. These are estimated from training samples.

If the covariance matrix is singular, which is possible in high feature space, we may need to approximate the inverse by adding a small regularization term λ to the Σ [$\Sigma' = (1 - \lambda)\Sigma + \lambda I$).[7] When d is very large, approximating the inverse of Σ can be extremely computationally expensive and memory consuming. Therefore, a feature selection/extraction step is necessary beforehand.

2.1.2. Parzen Windows. Parzen windows is an extension of global density estimation and copes with multimodal class distribution. The estimated density is an average of a mixture of a Gaussian (in most cases) kernel centered on the individual training objects with a diagonal covariance matrix ($\Sigma = hI$):

$$p_{\mathrm{parzen}}(z) = \frac{1}{n}\sum_i p_N(z;\, x_i,\, hI)$$

where n is the number of training samples and h is the window width, which controls the sensitivity of the local density estimation.

2.2. Boundary Methods. *2.2.1. Nearest Neighbor Data Description (NNDD).* This method is an extension of the well-known nearest neighbor classifier.[6] The decision function of NNDD is written as

$$f_{\mathrm{NN}_{\mathrm{tr}}}(z) = I\left(\frac{||z - \mathrm{NN}^{\mathrm{tr}}(z)||}{||\mathrm{NN}^{\mathrm{tr}}(z) - \mathrm{NN}^{\mathrm{tr}}(z)||} < \text{threshold}\right)$$

where

$$I(\text{condition}) = \begin{cases} 1 & \text{if condition is true} \\ 0 & \text{if condition is false} \end{cases}$$

$\mathrm{NN}^{\mathrm{tr}}(z)$ means nearest neighbor to object z.

This function corresponds to a one-nearest neighbor classifier and can be easily extended to k-nearest neighbor by taking the average distance of k nearest neighbors. The value of rejection threshold can be set to 1 or defined by the user.

2.2.2. Support Vector Data Description (SVDD). There are two versions of a support vector machines (SVM) extension for one-class classification: SVDD[8] and ν-SVM.[9] These two SVMs are based on minimizing different structure errors: SVDD finds a minimum hypersphere that can cover most target samples, while ν-SVM separates target samples from the origin as well as possible. However, it can be proven that when these two SVMs are trained on the data set with unit variance or using a Gaussian kernel function, the solutions can be identical.[7] In this study, we use SVDD only.

In SVDD, the structure error is defined as

$$\epsilon(R, a, \xi) = R^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{with constraints } ||x_i - a||^2 \le R^2 + \xi_i$$

where R, a, and ξ are the radius, center of the radius, and the slack variable, respectively; x_i is a known object used for training; and n is the number of training samples. C is called the "penalty term"; it controls the size of ξ and, therefore, controls the fraction of training samples laying outside the boundary. See Figure 1 for an illustration of these three parameters.

This error function is then minimized using Lagrangian optimization. The final decision function for testing an unknown object z is written as

$$f_{\mathrm{SVDD}}(z; a, R) = I(||z - a||^2 \leq R^2) = I[(z{\cdot}z) - 2\sum_{i=1}^{n}\alpha_i(z,x_i) + \sum_{i,j=1}^{n} a_i a_j(x_i,x_j) \leq R^2]$$

where

$$a = \sum_{i=1}^{n}\alpha_i x_i \text{ and } R^2 = (x_k,x_k) - 2\sum_{i,j=1}^{n}\alpha_i\alpha_j(x_i,x_k)$$

α_i is the Lagrangian multiplier; for the nonsupport vectors objects, it always equals 0.

In this function, (x_i,x_j) represents an inner product between two objects x_i and x_j. SVDD using an inner product can only yield a rigid hypersphere boundary, which rarely fits data well in practical situations. Analogous to its counterpart in binary classification, one can replace this inner product (x_ix_j) with other kernel functions $K(x_i,x_j)$ to get a flexible boundary. In practice, a Gaussian kernel (also known as a radial-based function, RBF) $K(x_i,x_j) = \mathrm{e}^{-||x_i-x_j||^{2/\sigma}}$ is the most suitable kernel function for one-class classification. When using this kernel function, there is a kernel parameter σ that must be specified by the user. When σ changes from small to large values (compared to the numerator term in the kernel function), the decision boundary changes from a Parzen window such as local density estimation with a very small h to a rigid hypersphere[8] (from overfitting to overgeneralization). In this study, both of these types of SVDD are considered and compared with other OC classifiers.

2.3. Reconstruction Methods. *2.3.1. K-Means Data Description (K-means-DD).* In *K*-means-DD,[7] one first specifies a number of clusters k and then performs k-means clustering[10] on the training samples. After clustering, k centers (prototypes) are found; each prototype represents one cluster. Samples are assigned to the cluster with the shortest distance to its prototype. Therefore, the target class is represented by those prototypes. When testing an unknown test sample z, distances from z to each prototype are calculated and the minimum one (construction error) is taken for decision. To avoid some erroneous local minima, one may need to run k-means several times with different initial prototype guesses and retain the one with the minimum k-means error:

$$\epsilon_{k-m} = \sum_{i=1}^{n}\min_k||x_i - u_k||^2$$

where n is the number of samples and k is the number of clusters.

2.3.2. PCA Data Description (PCA-DD). PCA-DD can be considered as a counterpart of the well-known soft independent modeling of class analogies (SIMCA)[11] in multiclass classification. They share the same methodology, and both are based on principal component analysis (PCA), a well-known technique for data reconstruction. PCA projects a d-dimensional matrix into a few orthogonal subspaces (called principal components, PCs) that describe the variance of data as well as possible. Therefore, the original data matrix $\mathbf{X}$ is decomposed into a product of two smaller matrices: a scores matrix ($\mathbf{T}$) and a loadings matrix ($\mathbf{P}$)

$$\mathbf{X} \approx \mathbf{T}{\cdot}\mathbf{P}$$

When PCA is used for one-class classification, one first builds a PCA model on training samples, and the construction error of an unknown testing object z is defined as the squared Euclidean distance between z and its PCA modeled version z':

$$\epsilon_{\mathrm{PCA}}(z) = ||z - z{\cdot}\mathbf{P}^{\boldsymbol{T}}{\cdot}(\mathbf{P}{\cdot}\mathbf{P}^{\boldsymbol{T}})^{-1}{\cdot}\mathbf{P}||^2 = ||z - z{\cdot}\mathbf{P}^{\boldsymbol{T}}{\cdot}\mathbf{P}||^2$$

The number of PCs should be specified by the user. It can be determined by cross-validation[12] or by simply using sufficient PCs that explain a high enough variance of the data.

2.4. Considerations for One-Class Classification. In one-class classification, there are two types of error:

Type I (ϵ_{I}): A target sample falsely being rejected (false negative).

Type II (ϵ_{II}): An outlier sample falsely being accepted (false positive).

The relative size of these of errors can be changed by setting different thresholds. To get a clear view of how these two types of error change along with the threshold, we use a receiver operator characteristic (ROC) curve.[13] To plot a ROC curve, we change the threshold value from high to low. Different pairs of Type I and Type II error are obtained, and by plotting Type II errors against those of Type I, we get a ROC curve. The best combination of these two types of error is on the top-left region of the ROC curve, which indicates the best performance that a classifier can achieve. A ROC curve gives a clear view of the behavior of a classifier, but comparing two ROC curves (i.e., comparing two different classifiers) is sometimes difficult. Therefore, we can integrate the area under a ROC curve (AUC) [14] using

$$\mathrm{AUC} = \sum_{i=1}^{n}\left\{(1 - \epsilon_{\mathrm{I}}^{i}{\cdot}\Delta\epsilon_{\mathrm{II}}) + \frac{1}{2}[\Delta(1 - \epsilon_{\mathrm{I}}){\cdot}\Delta\epsilon_{\mathrm{II}}]\right\}$$

where $\Delta\epsilon_{\mathrm{II}} = \epsilon_{\mathrm{II}}^{i} - \epsilon_{\mathrm{II}}^{i-1}$, $\Delta(1 - \epsilon_{\mathrm{I}}) = (1 - \epsilon_{\mathrm{I}}^{i}) - (1 - \epsilon_{\mathrm{II}}^{i-1})$, and n is the number of samples required to get an overall performance indicator of that model. The value of AUC typically varies from 0.5 to 1. A value of 1 indicates a perfect discrimination, and 0.5 indicates no separation at all. For some methods for which the threshold is predefined (e.g., SVDD), we can use the net output of a decision function (the value of the left part of a decision function) instead. It is usually not necessary to integrate the entire ROC region; the region most interesting to us is when the Type II error is lower than 0.5; therefore, we only integrate the region from 0.05 to 0.5, and the integrated AUC is then normalized by 0.45 to rescale onto the range 0−1.

Although ROC and AUC show the discriminating capability of an OC classifier, it does not mean that the classifier can achieve the optimum performance in the ROC curve in practical situations. Because most OC classifiers do not automatically give an optimum decision threshold, we have to estimate this threshold from training samples. If sufficient samples are available, we can determine this by using an independent validation set. But in gene-expression profile analysis, the number of samples is usually very limited. A feasible practice is to determine the threshold on the training set itself. By specifying the fraction of training samples that

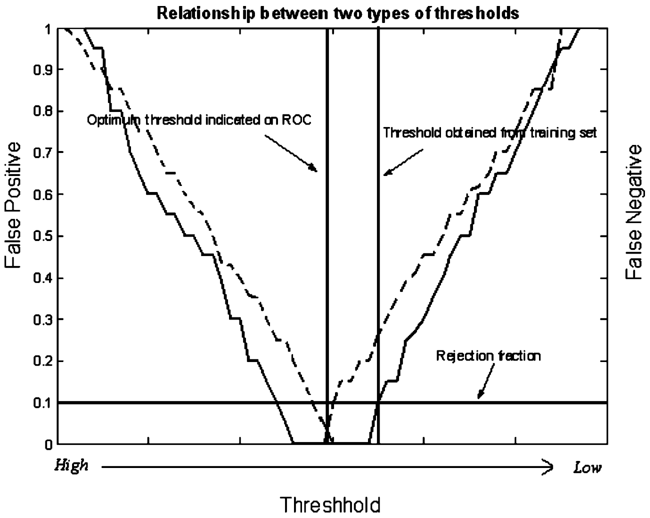

Figure 2. Relationship between the threshold indicated in ROC and the threshold found from the training set. The error curve on the left-hand side indicates Type II error, and that on the right-hand side indicates Type I error. The dashed line represents the error in blind test samples, and the solid line represents the error curve in the training samples. The value of the decision threshold is varied from high to low; if too high, a large Type II error is obtained, and if too low, a large Type I error is obtained. The optimum threshold is obtained when these two error curves cross (in ROC, indicated by the upper-left corner of the curve). Unfortunately, during the training stage, usually no outlier information is available and one has to arbitrarily set a threshold by excluding a small fraction of training samples. In this figure, the fraction is set to 10% (0.1 in the *y* axis). However, the error curve of the test set could be quite different (dashed line vs solid line) and, therefore, could incur a higher error rate than it is supposed to have.

should be rejected (e.g., 10%), we can set the threshold to the one that rejects this desired proportion of training samples (SVDD is an exception; the threshold is predefined by a squared radius, which is always obtained from training samples). Sometimes the training samples poorly represent the data, in which case ROC and AUC are actually overoptimistic performance indicators. The relationship between ROC and the actual performance of a one-class classifier is illustrated in Figure 2.

2.5. Feature Selection. For some one-class methods such as the Gaussian model, feature selection is essential if there are too many features. Gene-expression profile data usually have thousands of features, but there are relatively few samples, and a large number of the features are uninformative. So, we expect feature selection is not only imperative for those methods needed but also beneficial for those ones that can work on data sets with a lot of features. Unfortunately, it is difficult to do feature selection by one-class classification itself because of a lack of “outside” information. A large number of feature selection methods have been developed in supervised learning; most of these rely on a binary or multiclass classifier that separates a few predefined classes.[15] Some of the features selected by such methods may be only beneficial to the specific classifier but are not really biologically significant, and some interesting features may be left out since they contribute little to a specific separation. There is also a risk of eliminating some features that separate the target samples and the outliers. Ideally, features used for one-class classification should “naturally” relate most closely to the target class rather than contributing to a boundary that separates predefined classes.

To achieve this objective, we employed an unsupervised technique called coupled two-way cluster analysis (CTWC).[16,17] It is an iterative technique that attempts to discover whether there are any clusters in the samples and, if there are, which variable cluster(s) are most correlated to which sample cluster.

The algorithm starts by performing a cluster analysis on both samples and genes (variables) simultaneously. (When clustering samples, genes are used as variables, and when clustering genes, samples are used as variables.) Theoretically, any valid clustering algorithm could be used here. We use a new proposed hierarchical clustering algorithm called “super-paramagnetic clustering” (SPC) proposed by Blatt et al.[17] Compared to conventional hierarchical cluster analysis, it shows better performance on large data sets and irregular cluster shapes. A number of stable sample clusters and also gene clusters are obtained by SPC. The term “stable” indicates that the cluster has a considerable number of samples (in this study, ≥ 15 for genes and ≥ 5 for samples) and also is clearly separated from neighboring clusters. In the dendrogram, such separation is represented as the difference between the distance (*y* axis) when the cluster starts to emerge and the distance when the cluster is merged to a neighboring cluster. Sample clusters are denoted as S_1, S_2, ..., S_m, and gene clusters are denoted as $G_{1,}$ G_2, ..., G_n, providing *m* sample clusters and *n* gene clusters are recovered. Then, SPC is performed again on each sample−gene cluster pair, for example, S_1 on G_1, G_2, ..., G_n and also G_1 on S_1, S_2, ..., S_m and so on. Another set of gene clusters and sample clusters may be generated, and this procedure keeps repeating on new generated cluster pairs until no stable clusters can be found or the desired depth, that is, the number of iterations, is reached (no more than three because of computational complexity).

The output of CTWC is a series of sample clusters and gene clusters. To find out which gene clusters might be of interest, we calculate two indices called “purity” and “efficiency” on each sample cluster:

$$\text{purity}(s|c_i) = \frac{|s \cap c_i|}{|s|}, \qquad \text{efficiency}(s|c_i) = \frac{|s \cap c_i|}{|c_i|}$$

where $|s|$ is the number of samples of one cluster identified by CTWC, $|s \cap c_i|$ is the number of samples in that cluster belonging to known class *i*, and $|c_i|$ is the total number of samples belonging to class *i*. In simple terms, purity represents the diversity of one cluster: if one cluster only consists of samples from one known class, the purity of this cluster is 1. Efficiency represents the fraction of samples belonging to the same known class being recovered by the clustering algorithm and assigned to the same cluster. If one sample cluster obtained high values of both purity and efficiency on class *i*, the corresponding gene cluster should be the variable cluster that most correlated to that class. Since the whole procedure is done in an unsupervised way, the gene clusters recovered could be considered as “naturally” related to corresponding sample groups.

3. DATA SETS AND SOFTWARE

We used three data sets in this study. They are the SRBCT (small-round-blue-cell-tumor) data set,[1] the Colon data set,[2] and the Leukemia data set.[3]

Table 1. Data Sets

data set	number of samples	number of features
SRBCT	29(EWS), 11(BL), 18(NB), 25(RMS), 5(non-SRBCTs)	2308
Colon	40(cancer), 22(normal)	2000
Leukemia	47(ALL), 25(AML)	7129

The SRBCT data set consists of four subclasses of SRBCT, named the Ewing family of tumors (EWS), Neuroblastoma (NB), Burkitt lymphoma (BL), and Rhabdomyosarcoma (RMS). A total of 88 samples is available, described by 2308 genes. Of the samples, 29 are EWS, 11 are BL, 18 are NB, and 25 are RMS; five non-SRBCT samples are also included. The Colon data set has 62 samples described by 2000 genes. A total of 40 samples are colon cancer tissues, and 22 samples are normal tissues. The Leukemia data set has 72 samples, described by 7129 genes. Those samples correspond to variants of leukemia, named acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL); a total of 25 samples are AML and 47 are ALL. These data sets are summarized in Table 1.

We use a simple data preprocessing technique: first, normalize each sample to make the sum of the squares of each sample equal to 1; then, subtract the mean of each feature, and then, divide by the corresponding standard deviation.

All calculations are performed on Matlab 12.1. The Data-Description toolbox[18] combined with PRTools[19] is used for one-class classification. Other calculations are all performed on Matlab using in-house software.

4. RESULTS AND DISCUSSION

4.1. Parameter Settings. Apart from the Gaussian model and NNDD, the other four methods require one or more parameters to be specified by the user. These parameters may influence the performance of the model significantly. Parzen windows has one adjustable parameter: window width h. We optimize h by using a leave-one-out maximum likelihood solution proposed by Kraaijveld and Duin.[20] K-means has one adjustable parameter: the number of clusters k. We use a cluster validation step for choosing k. In cluster validation, the average silhouette width[21] is used as a validation index and $k_{\max}$ is set to the smallest integer larger than the square root of the number of samples. Sometimes, all silhouette widths are very low (e.g., < 0.2) and close to each other; this indicates that the clustering trend of the data is either very weak or being hidden because of the small number of samples and the high number of features. In such cases, k is set to half of $k_{\max}$. To avoid some erroneous local minima, we also repeated k-means clustering 10 times with different random initial guesses and retained the one with the minimum k-means error (ϵ_{k-m}).

SVDD has two adjustable parameters when using the Gaussian kernel function: C and σ. C can be optimized by the algorithm itself by specifying a fraction of target samples to be rejected, but σ has to be set to a suitable value. Considering the kernel function, we used

$$K(x_i, x_j) = \mathrm{e}^{-||x_i - x_j||2/\sigma}$$

given $x_i \neq x_{j.}$ If σ is too small, the exponential term would become very large and the output of the kernel function would always be 0 for any x_i and $x_{j.}$ If σ is too big, the exponential term will approach 0 and the output of the kernel function is always 1. In both cases, SVDD will give identical output for any test sample regardless of its origin, which means it loses its discriminant capability completely. The latter situation is usually not of concern, since σ usually needs to be unreasonably large. But, the first situation is very likely to occur when there are many features or the scale of the features is much bigger than σ as $||x_i - x_j||^2$ is an unnormalized term and is influenced by both the dimensionality of the data set and the scale of the features. Therefore, it has to be rescaled by an appropriate σ.

A good setting of σ should be adapted to the scale and number of features. In this study, we estimate the starting point of σ using the following equation:

$$\sigma = \frac{\sum_i ||x_i - \bar{x}||^2}{n}$$

where $\bar{x}$ is the mean of the whole data set (including the target class and the outlier class) and n is the number of samples.

Then, σ is optimized in a region of ±10-fold of the starting point using a grid search with a step of 0.1 × starting point. The σ that can yield the minimum fraction of samples that are support vectors (SV) is considered as the optimum. In leave-one-out cross-validation (LOO CV), there can be three kinds of situations: (1) If the sample being left out is not a SV, the solution obtained is the same and that sample can surely be correctly predicted. (2) If the sample being left out is a nonessential SV, the boundary will not change and it can still be correctly predicted. (3) If the sample being left out is an essential SV, the boundary will shrink and it will be rejected. So we can conclude that the LOO CV error is usually less than the fraction of SVs. It is worth noting that using a minimum fraction of samples that are SVs may lead to an extra large σ since the number of SVs is ultimately minimized when a spherical boundary is obtained. Therefore, the range of optimization has to be constrained within a small region around a sensible starting point.

PCA needs to know the number of significant factors beforehand. It is not suitable for a fixed number of PCs, since different classes may be characterized by different numbers of factors. Therefore, we simply retain enough PCs to explain at least 80% of the variance.

4.2. Feature Selection. First, each data set is randomly split into two parts: part I has 80% of the samples and is used by CTWC to identify the gene cluster of interest, and part II comprises the remaining 20% of the samples, used for testing the results of feature selection (see below).

In the Colon data set, we found two gene clusters that can yield very high purity and efficiency for normal and cancer classes. The first cluster comprises 45 genes; the value of the purity index for the normal class is 0.9, and for the cancer class, it is 0.8; the efficiency index for the normal class is 0.8, and for the cancer class, it is 0.7. The second gene cluster comprises 15 genes and yields a purity index of 0.9 and 0.8 for the normal and cancer classes, respectively. The efficiency index is 0.8 for both classes. So, in this data

Table 2. Significant Gene Clusters Found in the SRBCT Data Set[a]

	SRBCT							
	EWS		BL		NB		RMS	
number of genes	purity	efficiency	purity	efficiency	purity	efficiency	purity	efficiency
15	0.7	0.7	0.6	0.7	0.7	0.6	0.6	0.7
47	0.8	0.9						
58			1.0	1.0				
23					0.9	0.8		
50							1.0	0.8

[a] For individual gene clusters, except for the target samples, which are well clustered; other samples are either mixed together or form a series of unstable clusters (e.g., single sample cluster); the purity and efficiency of these clusters are omitted.

set, these two clusters, containing 45 + 15 = 60 genes, are used for OC modeling.

In the Leukemia data set, we found only one gene cluster, consisting of 60 genes that can yield high purity and efficiency; both purity and efficiency are 0.9 for the ALL class, while it is 0.9 and 0.8 for the AML class. Therefore, only this gene cluster is used for OC modeling.

In the SRBCT data set, we found that the situation is more complex. There is a small gene cluster (15 genes) that can yield reasonably high purity and efficiency for all four classes (only 0.6−0.7). But, we found several gene clusters that are only useful for modeling one class. For example, we found a gene cluster, having 58 genes, which, when samples are clustered on this subset of genes, obtains a pure BL cluster, and the purity and efficiency are both 1.0. It means all BL samples can be unambiguously clustered together without error, but other samples either cannot form a stable cluster or falsely cluster together. The other three classes also have such gene clusters. This is an encouraging sign: feature selection is focused on individual classes rather than an overall separation among several classes, and that is what OC modeling really needs. Therefore, when we perform the OC modeling, each class uses its own "individual gene cluster", a small gene cluster which can separate all four classes and is also used for each class. See Table 2 for a short description of these gene clusters.

To see whether feature selection can improve the results, we build OC classifiers on part I samples. OC classifiers are built for each class. Part II samples are used as the test set. When the samples belonging to one class are used as a target class, the remaining samples are considered as outliers. The performance of OC classifiers on the full feature set and subset selected by CTWC are compared in terms of the AUC. From these results, we find that feature selection can significantly improve the performance of the OC models (see Figure 3−5).

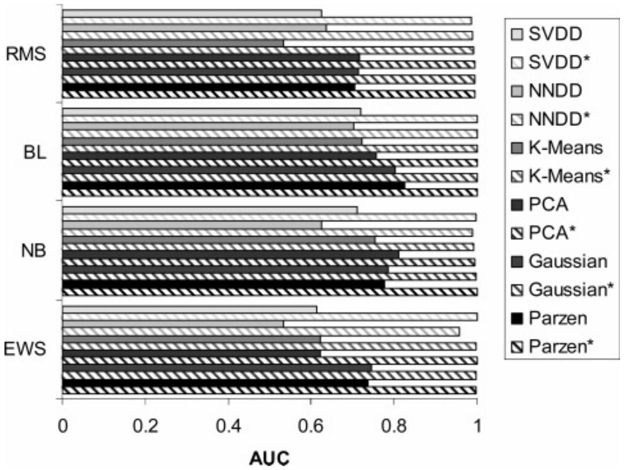

Figure 3. AUC improvement by feature selection on SRBCT data set.

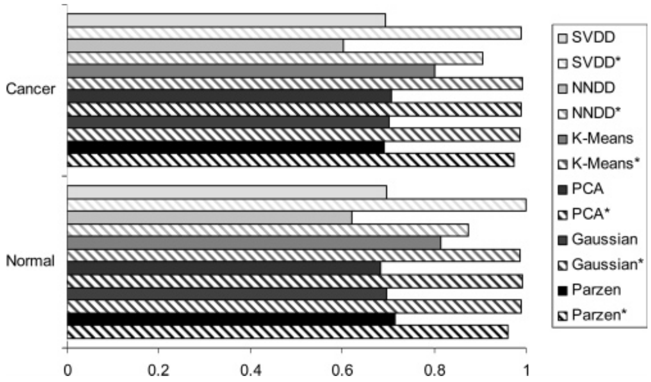

Figure 4. AUC improvement by feature selection on Colon data set.

4.3. Performance of OC Classifiers. It is important to realize that OC classifiers rely more on the representativeness of the training set compared to multiclass classifiers. A different split of training and test samples may yield significantly different results. We expect the sampling scheme can influence the performance of classifiers in two ways: the threshold estimated from the training set and the model free parameters estimated by the classifiers (e.g., support vectors in SVDD, μ and Σ in Gaussian model). The first aspect only influences two types of error (i.e., it will not significantly affect the extent of overlapping between the net outputs of a classifier for the target and outlier classes; the discriminating power of the classifier is not changed), whereas the second one will also influence the AUC (i.e., discriminating power of one classifier).

To get a view of "generalized" performance, we use a bootstrap[22] to evaluate the quality of the OC classifiers. Given n samples available in a data set, we randomly sample n samples with replacement for training, and those samples remaining unsampled are used for testing. In the training set, OC classifiers are trained on each class, and in the test set, while one class is used for the target class, the remainder are used for the outlier class. The decision threshold is obtained by rejecting 5% of the training samples. This bootstrap procedure is repeated 200 times; each time the AUC and two types of error of each classifier are calculated from the test set. Each classifier is built on the data set with

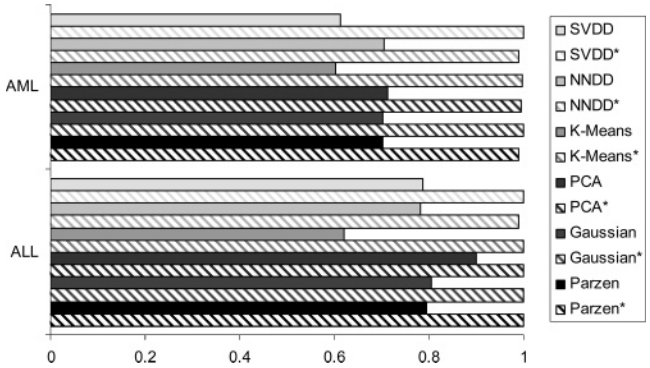

Figure 5. Performance improvement by feature selection on Leukemia data set.

its own optimum subset of features. We expect such a resampling strategy to cover many possible splitting schemes between training and test sets, from the most ideal conditions to very unsuitable conditions. The minimum and maximum errors (and AUC) represent each OC classifier's performance on those extreme conditions. The average errors and AUC can be considered as expected performance of those OC classifiers under "general" conditions. The standard deviation of those errors and the AUC is an indicator for the sensitivity to different splitting schemes. The errors are represented in Tables 3−5a; the AUCs are in Tables 6−8.

From the result, we can see that a different split of training and test samples can indeed yield significantly different results. In some cases, both types of error can vary from 0 to 100%. The standard deviation of the two types of error is also very high, from 0.1 to 0.35. They are presented in Tables 3−5a. This indicates that a well-sampled training set is crucial for the OC classifier. As for the AUC, most classifiers can achieve a high average AUC (close to 1), with the standard deviation varying from 0 to 0.2. The average of these two performance indicators is very consistent: a high average AUC usually corresponds to low Type I and Type II errors and vice versa.

Parzen windows failed on most data sets. They have a strict criterion that rejects almost every unknown sample, resulting in very high Type I error but zero Type II error. Even after we set the rejected fraction of the training set to 0%, the performance does not show any improvement. However, using the AUC criterion, Parzen windows is still a very powerful discriminator. In every dataset, the AUC of Parzen windows can reach 1 under the best conditions, which means a perfect discrimination in its net output. The problem is that we cannot obtain the optimum threshold by using the training set only, even in the best split of the training and testing sets. If we have more samples at hand, tuning the threshold by using a validation set may improve the performance of this classifier significantly. The Gaussian model performs quite well on the SRBCT data set but is not that successful on the Colon and Leukemia data sets. It indicates that in the SRBCT data set, four subclasses are all unimodal and can, therefore, be well-modeled by Gaussian functions. However, in the Colon data set, the Type II error is quite high compared to the Type I error for both the normal and cancer classes. This may be caused by the non-unimodality. In the Leukemia data set, the situation reversed: it becomes an "over-strict" classifier with high Type I and zero Type II errors. However, using the AUC criterion, we can see the Gaussian model actually has good discriminating power. The problem is that it is difficult to get an optimum decision threshold from limited training samples. The Gaussian model is also very sensitive to the sampling scheme, because two basic model parameters need to be estimated from training samples: μ and Σ. If training samples poorly represent the data, the decision boundary drawn by the Gaussian model can be completely wrong. We notice that in the Colon and Leukemia data sets, the AUC of the Gaussian model sometimes can be much lower than 0.5, which means the decision boundary covers more area in the outlier class than in the target class. That is caused by the poor estimation of μ and Σ.

SVDD is also a very strict classifier. The average Type I error is usually the second highest to Parzen windows, whereas the Type II error is usually very low. That is because SVDD always needs a certain number of samples to support the boundary (support vectors). As we have shown before, the estimated Type I error obtained by LOO CV is always less than or equal to the fraction of support vectors. If the size of the training set is small, the proportion of support vectors is usually large, resulting in a high Type I error. However, in some very good circumstances, Type I errors can still be reduced to 0, indicating all test samples are drawn from the inside of the boundary. This also indicates that given a well-sampled training set, even if the size is limited, SVDD can still achieve good performance. Comparing the two kinds of kernel functions we used, the inner product and the Gaussian, we found that the inner product usually has a higher Type II error than the Gaussian kernel. This may be caused by a rigid hypersphere boundary, which may not fit irregular-shaped data clusters well. When classes are less well-separated, we may get a high Type II error. NNDD is, on the contrary, a "loose" classifier. The Type I error is usually low, whereas the Type II error is high. It is also very sensitive to the split of training and testing samples. Type I and II errors can vary from 0 to 100%; it also shows high standard deviation in AUC, which indicates its discriminating power is also heavily reliant on sampling schemes. To achieve a good performance, a high quality of training samples is essential.

The performances of the two reconstruction-based methods *K*-means and PCA are usually midway between "over-strict" and "over-loose". Comparing these two classifiers, PCA is generally more "strict" than *K*-means. PCA is also the classifier least sensitive to the sampling scheme. The AUC of PCA shows the lowest standard deviation in most data sets. This is a feature of certain "extrapolating" capability PCA models. Instead of drawing a close boundary around target samples, PCA finds a series of linear subspaces that fit data points well. The difference between an unknown sample to its modeled version in those subspaces (construction error) is then used for the decision. Providing clear linear subspaces exist, PCA can still generalize well, even when training samples are limited. However, it still needs good training samples for the decision of the threshold and also for finding the most suitable subspaces. Moreover, *K*-means is more sensitive to the sampling scheme, since prototypes found by *K*-means are also purely dependent on training samples. The AUC can also be under 0.5 when training samples are poorly sampled.

Table 3. Two Types of Errors in the Colon Data Set

(a) By Decision Threshold

	false negative (FN)%							false positive (FP)%						
	SVL	SVG	NN	GA	PA	KM	PCA	SVL	SVG	NN	GA	PA	KM	PCA
Normal Class														
min	16.7	25	0	0	100	0	0	12.5	0	0	0	0	0	0
max	100	100	42.9	66.7	100	75	100	62.5	11.1	100	100	0	72.7	11.1
mean	61.7	64.1	4.1	8.3	100	16.3	49.0	29.3	0.1	49.1	56.1	0	20.1	3.1
std × 100	21.9	16.9	8.9	10.1	0	14.7	21.7	13.9	1.1	20.9	18.9	0	21.1	1.9
Cancer Class														
min	11.1	11.1	0	0	100	0	0	11.1	0	0	0	0	0	0
max	72.7	75	27.3	40	100	55.6	70	55.6	0	100	100	0	77.8	12.5
mean	45.1	59.7	5.9	6.8	100	13.3	34.1	28.7	0	48.5	59.2	0	14.5	2.1
std × 100	15.7	12.6	7.1	9.1	0	12.1	15.1	10.8	0	24.3	22.2	0	13.2	1.5

(b) By Clustering the Net Output of the OC Classifiers

	false negative (FN)%							false positive (FP)%						
	SVL	SVG	NN	GA	PA	KM	PCA	SVL	SVG	NN	GA	PA	KM	PCA
Normal Class														
min	0	0	0	0	0	0	0	12.5	0	0	0	0	0	0
max	100	100	40	75	100	66.6	100	55.6	12.5	100	100	0	87.5	9.1
mean	14.7	15.3	5.6	6.5	13.9	14.1	3.7	28.9	0.1	50.8	55.7	0	21.5	2.8
std × 100	13.9	14.6	7.9	9.1	8.7	18.6	6.1	12.4	0.9	21.4	17.4	0	23.4	1.8
Cancer Class														
min	0	0	0	0	0	0	0	12.5	0	0	0	0	0	0
max	75	75	25	33.3	100	55.6	66.7	66.7	0	100	100	0	75	12.5
mean	9.8	11.3	3.6	4.8	12.8	11.9	2.9	25.4	0	59.7	57.7	0	13.2	1.7
std × 100	10.1	13.4	6.1	7.9	8.5	9.5	4.1	8.8	0	23.9	20.9	0	11.9	0.6

Table 4. Two Types of Errors in the Leukemia Data Set

(a) By Decision Threshold

	false negative (FN)%							false positive (FP)%						
	SVL	SVG	NN	GA	PA	KM	PCA	SVL	SVG	NN	GA	PA	KM	PCA
AML Class														
min	0	0	0	0	100	0	0	12.5	0	0	0	0	0	0
max	100	100	55.6	100	100	77.8	88.9	44.4	0	100	0	0	87.5	44.4
mean	59.8	65.9	6.6	74.9	100	14.3	26.4	24.5	0	69.2	0	0	47.2	1.6
std × 100	26.3	18.1	10.1	15.1	0	17.1	19.7	13.1	0	27.3	0	0	33.4	4.9
ALL Class														
min	0	0	0	0	37.5	0	0	12.5	0	0	0	0	0	0
max	60	66.7	33.3	66.7	100	44.4	62.5	75.0	0	37.5	0	0	14.3	0
mean	25.7	29.8	4.5	20.7	81.9	7.4	10.6	14.7	0	6.1	0	0	0.5	0
std × 100	16.8	11.9	6.9	12.7	10.1	8.9	10.1	9.1	0	12.1	0	0	2.9	0

(b) By Clustering the Net Output of the OC Classifiers

	false negative (FN)%							false positive (FP)%						
	SVL	SVG	NN	GA	PA	KM	PCA	SVL	SVG	NN	GA	PA	KM	PCA
AML Class														
min	0	0	0	0	0	0	0	9.1	0	0	0	0	0	0
max	100	100	66.7	100	100	75	100	50	0	100	0	0	100	14.3
mean	5.1	6.2	4.3	9.7	7.1	11.5	3.5	23.7	0	67.2	0	0	45.7	1.4
std × 100	9.3	10.2	7.4	10.9	10.1	13.2	4.8	12.5	0	28.6	0	0	34.5	3.8
ALL Class														
min	0	0	0	0	0	0	0	0	0	0	0	0	0	0
max	66.7	75	37.5	50	100	44.4	66.7	66.7	0	50	0	0	16.7	0
mean	0.4	0.5	2.9	0.1	0.5	1.3	0.3	10.4	0	5.9	0	0	0.4	0
std × 100	0.6	0.7	3.3	0.3	0.8	1.2	0.5	8.7	0	13.7	0	0	2.4	0

Although two types of errors are generally consistent with AUC, we find there is still a significant “gap” between these two kinds of performance indicators. Sometimes, certain classifiers can result in a very high average AUC but still have high averages for both types of errors. This indicates that, sometimes, although our classifiers are well-trained and there is a good separation between the net outputs (i.e., numerical output of the classifier rather than predicted label) of the target and outlier samples (AUC close or equal to 1), estimating the threshold purely from training samples prevents the classifier from achieving such good separation on unknown test samples. However, there could be a way to work around such a problem. Instead of using a threshold to make the decision, we can perform a cluster analysis on the net output of the OC classifier. Given a well-trained classifier, the net outputs of the target samples are always

Table 5. Two Types of Errors in the SRBCT Data Set

(a) By Decision Threshold

	false negative (FN)%							false positive (FP)%						
	SVL	SVG	NN	GA	PA	KM	PCA	SVL	SVG	NN	GA	PA	KM	PCA
							EWS Class							
min	0	12.5	0	87.5	75	0	0	0	0	0	0	0	0	0
max	87.5	100	44.4	100	100	66.7	87.5	25	0	100	0	0	22.2	9.1
mean	41.9	50.1	5.1	91.7	92.3	11.5	39.2	13.4	0	51.3	0	0	1.3	0.01
std × 100	15.8	16.7	11.3	5.1	4.1	12.9	19.5	1.7	0	34.6	0	0	4.1	0.2
							NB Class							
min	0	0	0	100	100	0	0	0	0	0	0	0	0	0
max	100	100	75	100	100	100	100	12.7	0	50	0	0	0	0
mean	43.1	59.3	10.3	100	100	58.1	59.3	3.9	0	6.2	0	0	0	0
std × 100	30.9	27.6	18.1	0	0	31.6	23.0	5.9	0	10.1	0	0	0	0
							BL Class							
min	16.7	25	0	66.7	100	0	0	0	0	0	0	0	0	0
max	66.7	87.5	50	100	100	66.7	66.7	25	0	72.7	0	0	0	0
mean	31.6	43.7	5.8	81.6	100	27.6	25.3	10.3	0	13.7	0	0	0	0
std × 100	16.8	15.1	8.5	5.9	0	17.9	16.9	13.9	0	25.6	0	0	0	0
							RMS Class							
min	0	0	0	0	75	0	0	0	0	0	0	0	0	0
max	66.7	75	44.4	100	100	87.5	66.7	25	0	72.7	9.1	0	25.0	12.5
mean	23.9	39.2	6.5	43.6	91.0	11.2	10.3	9.6	0	10.4	0.1	0	1.8	0.9
std × 100	11.5	16.8	9.7	18.7	5.2	15.3	9.9	2.7	0	12.9	0.7	0	2.1	1.8

(b) By Clustering the Net Output of the OC Classifiers

	false negative (FN)%							false positive (FP)%						
	SVL	SVG	NN	GA	PA	KM	PCA	SVL	SVG	NN	GA	PA	KM	PCA
							EWS Class							
min	0	0	0	0	0	0	0	0	0	0	0	0	0	0
max	100	100	33.3	100	100	75	66.7	12.5	0	100	0	0	25.0	11.1
mean	2.5	4.1	2.6	2.4	2.8	5.9	3.7	4.1	0	49.2	0	0	1.5	0.01
std × 100	1.1	2.3	1.5	4.2	3.7	2.4	1.9	1.6	0	33.1	0	0	4.2	0.2
							NB Class							
min	0	0	0	0	0	0	0	0	0	0	0	0	0	0
max	100	100	100	100	100	100	100	10.0	0	44.4	0	0	0	0
mean	6.2	8.3	9.1	5.9	3.2	8.5	2.9	3.7	0	5.3	0	0	0	0
std × 100	4.4	4.6	5.1	4.1	3.8	6.7	3.9	1.5	0	9.9	0	0	0	0
							BL Class							
min	0	0	0	0	0	0	0	0	0	0	0	0	0	0
max	75	50.0	44.4	100	100	75	50	33.3	0	75.0	0	0	0	0
mean	4.7	5.6	5.8	4.6	3.9	4.6	3.9	11.3	0	11.7	0	0	0	0
std × 100	3.8	4.1	3.5	3.9	4.7	4.4	4.0	14.1	0	23.7	0	0	0	0
							RMS Class							
min	0	0	0	0	0	0	0	0	0	0	0	0	0	0
max	66.7	66.7	33.3	100	100	75	50	25	0	66.7	12.5	0	33.3	12.5
mean	3.9	4.8	2.7	6.9	7.2	3.1	2.9	9.1	0	8.5	0.1	0	2.1	0.8
std × 100	2.1	2.9	1.8	4.7	5.1	2.8	1.5	3.1	0	10.6	0.6	0	2.8	1.6

Table 6. AUC on the Colon Data Set

AUC	SVL	SVG	NN	GA	PA	KM	PCA
				Normal Class			
min	0.4970	0.5174	0.6014	0.3687	0.5997	0.4076	0.8007
max	0.9517	1	1	1	1	1	1
mean	0.8145	0.8978	0.9011	0.7993	0.8679	0.8911	0.9339
std	0.1103	0.1001	0.0998	0.1669	0.0927	0.1083	0.0211
				Cancer Class			
min	0.5514	0.8006	0.5091	0.4002	0.8019	0.7675	0.8769
max	0.9708	1	1	1	1	1	1
mean	0.8819	0.9199	0.8571	0.8107	0.9495	0.9394	0.9694
std	0.1099	0.0437	0.1074	0.1206	0.0499	0.0519	0.0329

Table 7. AUC on the Leukemia Data Set

AUC	SVL	SVG	NN	GA	PA	KM	PCA
				AML Class			
min	0.6201	0.7045	0.6139	0.5109	0.6614	0.4977	0.7749
max	0.9871	1	1	1	1	1	1
mean	0.8799	0.9079	0.8997	0.9767	0.8999	0.8876	0.9597
std	0.0571	0.0684	0.0763	0.0697	0.0975	0.1009	0.0794
				ALL Class			
min	0.8979	0.9814	0.9277	1	0.9876	0.9675	0.9991
max	0.9903	1	1	1	1	1	1
mean	0.9799	0.9989	0.9936	1	0.9996	0.9981	1.0000
std	0.0101	0.0091	0.0109	0	0.0023	0.0099	0.0001

small and constrained into a small region, whereas those of the outliers are always large but may not necessarily be close to each other. So it is possible to cluster the net outputs into two (or more) clusters: one "low cluster" and one or more "high clusters". The "low cluster" could be considered as a cluster of target samples, whereas the remaining samples are all outliers. We tested this method using the same procedure as described above. The results improved significantly; even Parzen windows is successful. An obvious improvement is that the minimum and average Type I errors have been reduced significantly. That makes "over-strict" classifiers perform better than "over-loose" classifiers in overall terms.

Table 8. AUC on the SRBCT Data Set

AUC	SVL	SVG	NN	GA	PA	KM	PCA
			EWS Class				
min	0.8901	0.9609	0.8593	0.9874	0.9904	0.8871	0.9145
max	1	1	1	1	1	1	1
mean	0.9814	0.9908	0.9845	0.9991	0.9997	0.9983	0.9989
std	0.0211	0.0171	0.0767	0.0034	0.0013	0.0216	0.0103
			NB Class				
min	0.9077	0.9974	0.8699	0.9896	0.9917	0.9899	1
max	1	1	1	1	1	1	1
mean	0.9791	0.9998	0.9987	0.9999	0.9998	0.9999	1
std	0.0101	0.0012	0.0203	0.0009	0.0003	0.0013	0
			BL Class				
min	0.8979	0.9976	0.9027	0.9876	0.9731	0.8979	0.9913
max	1	1	1	1	1	1	1
mean	0.9678	0.9997	0.9817	0.9997	0.9994	0.9991	0.9998
std	0.0217	0.0124	0.0616	0.0010	0.0079	0.0121	0.0006
			RMS Class				
min	0.8869	0.9793	0.8915	0.9123	0.9087	0.9799	0.9869
max	1	1	1	1	1	1	1
mean	0.9714	0.9975	0.9779	0.9899	0.9884	0.9981	0.9995
std	0.0357	0.0179	0.0321	0.0139	0.0207	0.0099	0.0086

The Type II error does not change much as the false positive error indicates the net outputs of the outliers are overlapped by those of the target samples and, therefore, cannot be separated by clustering. Also, the maximum Type I error of each classifier is still high, which indicates poor training set results in a poor classifier. The results are shown in Tables 3−5. The main drawback of this method is it requires the analysis to be performed in a "batch" mode; that is, both the target samples and outliers must be present in the data set and analyzed together.

5. CONCLUSION

In this study, we have demonstrated the feasibility of using one-class classifiers in gene-expression profile data sets for diagnostic purposes. The main advantage of these classifiers is they are robust to outliers. In diagnosis applications, this is very important. It is also easy to accommodate new classes without changing already existing OC classifiers. Six OC classifiers are compared; no classifier performs consistently better than the others. Despite the large number of features and limited samples, most OC classifiers can still achieve adequate performance providing there is a good splitting of the training and testing sets.

However, there are two main difficulties in applying those OC classifiers: a large number of features (many of which are irrelevant) and a limited number of samples. These two problems are common in gene-expression profile analysis, but OC classifiers are very sensitive to these problems. The first can be solved by using a feature selection step. We employ an unsupervised two-way clustering technique to solve this problem. It has the capability to identify a subset of features that most correlate to each individual class, which makes it especially suitable for OC applications. The limitation about the number of samples could be a serious problem. From the results we obtained, we can see that a well-constructed training set is crucial for OC modeling. If the training set is appropriate, the performance of the OC models is comparable to that of conventional multiclass classifiers. And as a result of their built-in outlier detection capability, they are more suitable for diagnostic purposes.

The setting of a decision threshold is also very important. For most binary (as well as multiclass) classifiers (e.g., LDA, SVM, etc.), this problem does not exist as samples are separated by the sign of the net output; that is, negative net output represents class 1 and positive net output represents class 2. But, the net output of OC classifiers is not well-defined. It can be any real number depending on the algorithm of the classifier and how it was trained. Therefore, it is very important to find out an optimum threshold that can maximize the true performance of the classifier. Given enough samples, we can build a probability density function of the net outputs of target samples and obtain the optimum threshold from it. If the sample size is limited, we find using cluster analysis on net outputs can yield better results than using a threshold.

REFERENCES AND NOTES

(1) Kahn, J.; Wei, J. S.; Ringner, M.; Saal, L. H.; Ladanyi, M.; Westermann, F.; Berthold, F.; Schwab, M.; Antonescu, C. R.; Peterson, C.; Meltzer, P. S. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Med.* **2001**, *7*, 673−679.

(2) Alon, U.; Barkai, N.; Notterman, D. A.; Gish, K.; Ybarra, S.; Mack, D.; Levine, A. J. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. U.S.A.* **1999**, *96*, 6745−6750.

(3) Golub, T. R.; Slonim, D. K.; Tamayo, P.; Huard, C.; Gaasenbeek, M.; Mesirov, J. P.; Coller, H.; Loh, M.; Downing, J. R.; Caligiuri, M. A.; Bloomfield, C. D.; Lander, E. S. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science* **1999**, *286*, 531−537.

(4) Hampel, F. R.; Ronchetti, E. M.; Rousseeuw, P. J.; Stahel, W. A. *Robust Statistics* − *The Approach based on Influence Functions*; Wiley: New York, 1986.

(5) Rousseeuw, P. J.; Van Driessen, K. A fast algorithm for the minimum covariance determinant estimator. *Technometrics* **1999**, *41*, 212−223.

(6) Duda, R. O.; Hart, P. E.; Stork, D. G. *Pattern Classification*, 2nd ed.; Wiley: New York, 2001.

(7) Tax, D. M. J. One class classification. Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 2001.

(8) Tax, D. M. J.; Duin, R. P. W. Support vector domain description. *Pattern Recognit. Lett.* **1999**, *20*, 1191−1199.

(9) Schölkopf, B.; Williamson, R.; Smola, A.; Shawe-Taylor, J. SV estimation of a distribution's support. In *Neural Information Processing Systems 99*; Kearns, M. S., Solla, S. A., Cohn, D. A., Eds.; MIT Press: Cambridge, MA, 1999.

(10) Everitt, B. S.; Landau. S.; Leese, M. *Cluster Analysis*, 4th ed.; Edward Arnold: London, 2001.

(11) Wold, S. Pattern Recognition by Means of Disjoint Principal Components Models. *Pattern Recognit.* **1976**, *8*, 127−139.

(12) Brereton, R. G. *Chemometrics: Data Analysis for the Laboratory and Chemical Plant*; Wiley: Chichester, U. K., 2003.

(13) Metz, C. E. Basic Principles of ROC Analysis. *Semin. Nucl. Med.* **1978**, *8*, 283−298.

(14) Bradley, A. P. The Use of the Area under the ROC Curve in the Evaluation of Machine Learning Algorithms, *Pattern Recognit.* **1997**, *30*, 1145−1159.

(15) Guyon, I. An Introduction to Variable and Feature Selection. *J. Machine Learning* **2003**, *3*, 1157−1182.

(16) Getz, G.; Levine, E.; Domany, E. Coupled two-way clustering analysis of gene Microarray data. *PNAS* **2000**, *97*, 12079−12084.

(17) Blatt, M.; Wiseman, S.; Domany, E. Super-paramagnetic clustering of data. *Phys. Rev. Lett.* **1996**, *76*, 3251−3254.

(18) Tax, D. M. J. *Data Description toolbox (dd_tools)*. Available at http://www-ict.ewi.tudelft.nl/~davidt/dd_tools.html.

(19) Ridder, D. *PRTools*. Available at http://www.prtools.org.

(20) Kraaijveld, M.; Duin, R. P. W. Technical Report; Delft University of Technology: Delft, The Netherlands, 1991.

(21) Rousseeuw, P. J. Silhouettes − A Graphical Aid to the Interpretation and Validation of Cluster-Analysis. *J. Comput. Appl. Math.* **1987**, *20*, 53−65.

(22) Efron, B. Bootstrap Methods − Another Look at the Jackknife. *Ann. Statistics* **1979**, *7*, 1−26.

CI049726V