Available online at www.sciencedirect.com
ScienceDirect
Neural Networks 21 (2008) 502–510
Neural Networks
www.elsevier.com/locate/neunet

2008 Special Issue

New support vector-based design method for binary hierarchical classifiers for multi-class classification problems☆

Yu-Chiang Frank Wang*, David Casasent

Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Received 3 August 2007; received in revised form 9 November 2007; accepted 3 December 2007

Abstract

We propose a new hierarchical design method, *weighted support vector (WSV) k-means clustering*, to design a binary hierarchical classification structure. This method automatically selects the classes to be separated at each node in the hierarchy, and allows visualization of clusters of high-dimensional support vector data; no prior hierarchical designs address this. At each node in the hierarchy, we use an SVRDM (support vector representation and discrimination machine) classifier, which offers generalization and good rejection of unseen false objects (rejection is not achieved with the standard SVMs). We give the basis and new insight into why a Gaussian kernel provides good rejection. Recognition and rejection test results on a real IR (infrared) database show that our proposed method outperforms the standard one-vs-rest methods and the use of standard SVM classifiers.

Keywords: Automatic target recognition; Hierarchical classifier; Pattern recognition; Support vector machine

1. Introduction

When the number of classes in a multi-class problem increases, computational complexity increases and performance can decrease. Non-parametric classifiers (e.g. kNN) are not attractive because of their computational complexity (Fukunaga, 1990). Since it is easier to construct a classifier for two true classes than one classifier to handle C different classes (Kumar, Ghosh, & Crawford, 2002; Platt, Cristianini, & Shawe-Taylor, 1999), a preferable method for the multi-class classification problem is to use several binary classifiers and to combine their results. Therefore, we consider a hierarchical classifier (see Fig. 1); at the top node, we decompose the C-class problem into two binary two-class (or macro-class) problems (*a macro-class is a collection of several classes*) in the hierarchy. At each node in the hierarchy, the input is classified into one of two macro-classes. The *design* of the hierarchy refers to selecting the macro-classes to be separated at each node. Among the binary classifiers, the support vector machine (SVM) is an attractive classifier that provides generalization (Cortes & Vapnik, 1995). In this paper, we consider a hierarchy of SVM-type classifiers (one at each node in Fig. 1). Our work also concerns rejection of unseen false objects. To achieve this, we use our new SVRDM (support vector representation and discrimination machine (Section 3)) classifier in the hierarchy. It provides an improved rejection ability, which the standard SVM classifier does not (Tax & Duin, 1999).

There are two standard approaches to construct and combine the results from different classifiers for a C-class problem. The first is the *one-vs-rest* method (Duda, Hart, & Stork, 2000), in which each classifier distinguishes one class from the other C-1 classes, and the class label of the input is decided by winner-take-all, etc., methods (Anand, Mehrotra, Mohan, & Ranka, 1995). Each classifier needs to be trained on the whole training set, and there is no guarantee that good discrimination exists between one class and the remaining classes. This method also results in imbalanced data learning problems, where the number of training set samples in the two classes is very different. Therefore, we may not expect good classification using this method. The second standard approach to combine different classifiers is the *one-vs-one* method (Hastie & Tibshirani, 1998), in which the decision is made by majority voting, etc.,

☆ An abbreviated version of some portions of this article appeared in Wang and Casasent (2007) as part of the IJCNN 2007 Conference Proceedings, published under IEE copyright.

* Corresponding author. Tel.: +1 412 268 4495.
E-mail address: ycwang@cmu.edu (Y.-C.F. Wang).

0893-6080/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2007.12.005

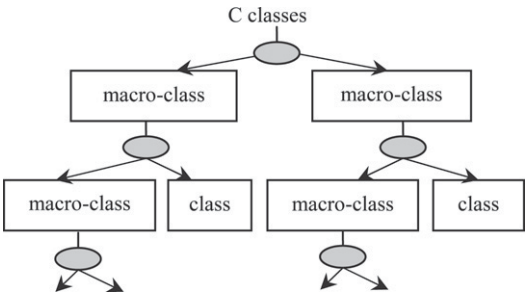

Fig. 1. Structure of a binary hierarchical classification structure.

strategies. This requires training and testing of $C(C-1)/2 \approx C^2/2$ different classifiers. This approach is prohibitive when C is large (Kumar et al., 2002).

Thus, we chose a binary hierarchical classification structure in Fig. 1 (Casasent & Wang, 2005; Wang & Casasent, 2006) using new SVRDM classifiers (Yuan & Casasent, 2003) for the multi-class problem. In Fig. 1, each node is a binary classifier. Coarse separation between classes occurs in the beginning (at upper levels) in the hierarchy and a finer classification decision later (at lower levels). At the top node, we divide the original C classes into two smaller groups of classes (*macro-classes*); this clustering procedure is repeated in subsequent levels, until there is only one class in the final sub-group. The macro-classes are different at each node in the hierarchy and the numbers of classes in each macro-class at a node are not necessarily the same. This hierarchical structure decomposes the problem into C-1 binary sub-problems. For testing, only about $\log_2 C$ classifiers are required to traverse a path from top to bottom. Thus, the number of required calculations is reduced in this approach.

We originally (Casasent & Wang, 2005) considered a balanced binary hierarchical structure, in which the two macro-classes at each node had approximately the same number of classes. We now propose a new clustering algorithm (Section 2.2), which designs the hierarchical structure (the macro-classes at each node) in the same high dimensional space, in which the SVRDMs are used. This method does not require a balanced structure.

In many pattern recognition applications, it is also necessary to provide good rejection of false classes that have never been seen in training, e.g. imposters in face recognition; this issue should be considered in the design of the classifier. Surprisingly, the classification–rejection problem has not been addressed in most prior work. The standard SVM has poor rejection. Our SVRDM classifier is designed to give good classification (of true classes) and rejection (of false classes not seen in training).

This paper contains much data and information not present in our IJCNN'07 Proceedings paper (Wang & Casasent, 2007). There is a complete discussion in Section 2 on how we analyze the data in higher dimensional space, and how we use this information to propose the new hierarchical design method. This is much longer than the text in our IJCNN'07 paper. Section 3 details the SVRDM classifier and provides new insight into reasons for its good rejection. The discussion on time complexity in Section 4 is not in the IJCNN'07 paper. Sections 5 and 6 describe the database and feature space used. New Section 7 ROC data is not present in the IJCNN'07 paper.

2. Binary hierarchical classification structure and its design methods

The hierarchical clustering (i.e. the macro-class selection) at each node in the hierarchy should not be done arbitrarily or by intuition. There are two different design approaches: agglomerative (bottom-up) and divisive (top-down) clustering (Duda et al., 2000). Since there is no guarantee that a good separation exists at each node in bottom-up design, we consider a top-down design for our hierarchical classifier. We discuss other prior design methods in Section 2.1 and introduce our new method in Section 2.2.

2.1. Prior methods

The *spherical shell* method (Vural & Dy, 2004) compares the class means to the total mean. Since it only uses a simple mean vector per class, it does not describe the data well when the distribution is complex.

Others (Platt et al., 1999) proposed a Directed Acyclic Graph to combine the results of different one-vs-one SVM classifiers in a hierarchical way. However, classification errors seem to occur, because each classifier is only trained to distinguish between two classes and inputs will be from all true classes. This method also needs $\sim C^2/2$ classifiers.

To design a binary hierarchy, one could perform an exhaustive search for the macro-class pair with the largest margin at each node. However, for a node with M classes, there are $2^M/2 - 1 \approx 2^{M-1}$ macro-class pair combinations to consider, which can become prohibitive when M is large. To avoid an exhaustive search of all macro-class choices at each node, we proposed a *balanced binary hierarchical structure* method (Casasent & Wang, 2005), in which the number of classes in each macro-class at each node was the same or different by only one. For a node with eight classes, there are only $0.5 \times C_4^8 = 35$ possible macro-class pair combinations, instead of $2^8/2 - 1 = 127$. In the hierarchical design, we still need to evaluate all possible choices of balanced macro-class pairs at each node. Obviously, requiring an equal number of classes in each macro-class at a node may not be optimal.

Standard k-means clustering has been applied to design a binary hierarchical structure (Vural & Dy, 2004). To divide a set of classes into two groups ($k = 2$) at each node, they used the class mean $\boldsymbol{\mu}_i$ to represent each class i and applied standard k-means clustering to divide these mean vectors into two groups, such that the total distance between the means $\boldsymbol{\mu}_i$ (of all classes in cluster C_j) to its cluster centroid $\boldsymbol{m}_j$ is minimized. The squared error to be minimized was

$$J = \sum_{j=1}^{2} \sum_{\mu_i \in C_j} \left\| \boldsymbol{\mu}_i - \boldsymbol{m}_j \right\|^2. \tag{1}$$

The use of class mean is not good when the data distribution is complex (as we noted earlier). To address this, we proposed

(Wang & Casasent, 2006) use of one representative vector for each class (a combination of its support vectors) in place of $\boldsymbol{\mu}_i$ in (1). This design method is fast, and the unbalanced design it produces is expected to perform better than a balanced design. However, we calculated the representative vector as a linear combination of the support vectors in the original data space and the macro-class choices were made in the original space. In this paper (Section 2.2), our new algorithm uses the higher-order classification space to select the macro-classes.

In our hierarchical classifier, the feature space used is the same for all classifiers in the hierarchy. In Shaik and Yeasin (2006), subspace methods are used to select different features to use at different levels in a binary hierarchical structure (for microarray gene data separation of tumor and normal samples). This is very different from our proposed binary hierarchical "classification" structure.

2.2. Our proposed method

We now address our new hierarchical design method (i.e. selection of the classes to be separated at each node in the hierarchical tree). Since we apply SVM-type classifiers at each node in the hierarchy, it is preferable to design the hierarchy in the same high dimensional transformed feature space. Therefore, $\boldsymbol{\mu}_i$ and $\boldsymbol{m}_j$ in (1) must be in higher-order transformed space. We propose a new WSV *(Weighted Support Vector) K-means Clustering* method that separates a set of classes into two groups in high-dimension space using all support vector information for each class.

There are two major concerns in our method. First, we need to find a new vector (for each class) in higher-order transformed space to use for $\boldsymbol{\mu}_i$; this vector will be the best representative of class i in this space. The second concern is how to implement the k-means clustering algorithm in (1) in that space? To address the first problem, we use the SVRM (support vector representation machine (Yuan & Casasent, 2003)) solution vector, which determines one discriminant function that represents one class well. To implement (1) in higher-order space, we expand (1) as

$$J = \sum_{j=1}^{2} \sum_{x \in C_j} \left\| \boldsymbol{\mu}_i - \boldsymbol{m}_j \right\|^2$$

$$= \sum_{j=1}^{2} \sum_{x \in C_j} \left(\left\| \boldsymbol{\mu}_i \right\|^2 + \left\| \boldsymbol{m}_j \right\|^2 + 2\boldsymbol{\mu}_i^{\mathrm{T}} \boldsymbol{m}_j \right). \quad (2)$$

In (2), we note that each term is a vector inner product (VIP). Therefore, when implementing (2) in higher-order space, only the VIP between different transformed data is required, and thus a kernel function can be used to compute the result without knowing the transform explicitly. Section 2.2.1 reviews the SVRM and the use of kernels; our new clustering method and the solution to (2) are presented in Section 2.2.2.

2.2.1. Support vector representation machine (SVRM)

We review the SVRM, as it is used to select the representatives for each class. The SVRM is a modified one-class SVM that solves the data domain description problem (Tax & Duin, 1999). It recognizes one class (of interest) in a high dimensional transformed feature space with a Gaussian kernel. Its solution vector $\boldsymbol{h}$ satisfies

$$\text{Min } \|\boldsymbol{h}\|^2/2, \text{ s.t. } f(\boldsymbol{x}_i) = \boldsymbol{h}^{\mathrm{T}} \Phi(x_i) \geq 1, \quad i = 1, \ldots, N, \quad (3)$$

where N is the number of training samples in the class of interest and $\Phi(\boldsymbol{x})$ is the training data projected onto a higher dimension space. Applying Lagrange multipliers and Quadratic Programming techniques, the solution vector $\boldsymbol{h} = \sum \alpha_i \Phi(\boldsymbol{x}_i)$, $i = 1$ to N, where $\alpha_i \geq 0$ are the Lagrange multipliers, and $\Phi(\boldsymbol{x}_i)$ are the transformed training set data. The data with non-zero α_i are the *support vectors* of this class; *the SVRM solution vector $\boldsymbol{h}$ is a linear combination of the support vectors of the class of interest*. As in SVM-type classifiers, $\boldsymbol{h}$ is not solved explicitly and only the α_i are known. Although the nonlinear transform Φ and the explicit form of $\Phi(\boldsymbol{x})$ are generally not available, with the use of a kernel function $K(\boldsymbol{x}, \boldsymbol{y})$ (Cortes & Vapnik, 1995), one can easily compute the *evaluation function* $f(\boldsymbol{x}) = \boldsymbol{h}^{\mathrm{T}} \Phi(\boldsymbol{x})$ in (3) for a test input $\boldsymbol{x}$ from the inner products of different $\Phi(\boldsymbol{x}_i)$, i.e. $K(\boldsymbol{x}, \boldsymbol{x}_i) = \Phi(\boldsymbol{x})^{\mathrm{T}} \Phi(\boldsymbol{x}_i)$, as we now show. Using the solution $\boldsymbol{h}$ in (3), the evaluation function in (3) becomes

$$f(\boldsymbol{x}) = \boldsymbol{h}^{\mathrm{T}} \Phi(\boldsymbol{x}) = \sum \alpha_i \left(\Phi(\boldsymbol{x}_i)^{\mathrm{T}} \Phi(\boldsymbol{x}) \right)$$

$$= \sum \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}). \quad (4)$$

From (4), we see that we only need to evaluate kernels K.

By minimizing $\|\boldsymbol{h}\|$ in the evaluation function $f(\boldsymbol{x}) = \boldsymbol{h}^{\mathrm{T}} \Phi(\boldsymbol{x})$ in (3), we minimize the decision region for the class of interest and thus achieve good rejection of inputs in other classes, which have never been seen. In testing, if $f(\boldsymbol{x}) \geq T$, we accept the input $\boldsymbol{x}$ as a member of the true class; if $f(\boldsymbol{x}) < T$, we reject the input as a false class. The threshold T is varied to obtain ROC (receiver operating characteristic) data. Throughout this paper, we assume that *no false class samples to be rejected are available during training*. This is realistic for many applications, since we cannot expect data from all possible false classes. Others also make this assumption (Chen, Zhou, & Huang, 2001). In the SVRM and our SVRDM (Section 3), we use a Gaussian kernel function $\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}\|^2/2\sigma^2)$. It is easy to evaluate the kernel function for an input $\boldsymbol{x}$ and a training sample $\boldsymbol{x}_i$. Thus, we can evaluate the vector inner product in the evaluation function $\boldsymbol{h}^{\mathrm{T}} \Phi(\boldsymbol{x})$ in (4). We have shown (Wang & Casasent, 2006; Yuan & Casasent, 2005) that this Gaussian kernel function choice is best when rejection is of concern; our new method (Yuan & Casasent, 2003) for selecting σ also makes this SVRM quite unique.

2.2.2. New WSV K-means clustering algorithm for binary hierarchical design

Our new method for binary hierarchical design is WSV *(weighted support vector) K-means Clustering*. This uses the SVRM solution vector $\boldsymbol{h}$ for each class as the best representative for that class in the high dimensional transformed space (Section 2.2.1). For a set of M classes at one node, we divide them into the best two clusters or macro-classes (C_1 and

C_2) using our new k-means clustering ($k = 2$) algorithm in higher-order space. Although the explicit form of $\Phi(\boldsymbol{x})$ and the solution vector $\boldsymbol{h}$ in high dimensional space are generally not available, use of kernel functions computes the VIPs between the transformed data, and allows use of the L2-norm to calculate the distance between different samples in high dimensional space. *This is very new*.

We now detail how to implement this method. We first replace $\boldsymbol{\mu}_i$ in (1) with the SVRM solution $\boldsymbol{h}_i$ for class i. The centroid $\boldsymbol{m}_1$ of cluster (macro-class) C_1 in (1) is now $\Phi(\boldsymbol{m}_1)$ in high dimensional space; it is the mean vector of the $\boldsymbol{h}_j$ for the classes in that cluster (macro-class), and it is iteratively updated during the clustering process. Therefore,

$$\begin{aligned}\Phi(\boldsymbol{m}_1) &= \frac{1}{N_{C_1}} \sum_{j=1, j\in C_1}^{N_{C_1}} \boldsymbol{h}_j \\ &= \frac{1}{N_{C_1}} \sum_{j=1, j\in C_1}^{N_{C_1}} \left(\sum_i \alpha_{ij} \Phi(\boldsymbol{x}_{ij}) \right), \end{aligned} \tag{5}$$

where N_{C1} is the number of classes grouped into cluster C_1. The form for $\Phi(\boldsymbol{m}_2)$ for cluster C_2 (second macro-class) is similar, and $N_{C1} + N_{C2} = M$ (the total number of classes to be separated). The $\boldsymbol{h}_i$ is a linear combination of the support vectors of class i. Since the form for $\boldsymbol{h}_i$ and for the nonlinear transform Φ are not known, we rewrite the distance from the vector $\boldsymbol{h}_i$ for each class i to the means $\Phi(\boldsymbol{m}_1)$ and $\Phi(\boldsymbol{m}_2)$ for the two clusters in terms of inner products $\Phi(\boldsymbol{x}_i)^{\mathrm{T}} \Phi(\boldsymbol{x}_j)$. For example, the distance between the SVRM solution vector $\boldsymbol{h}_i$ for class i and the mean $\Phi(\boldsymbol{m}_1)$ for cluster C_1 is

$$\begin{aligned}\|\boldsymbol{h}_i - \Phi(\boldsymbol{m}_1)\|^2 &= \left\| \sum_n \alpha_{in} \Phi(\boldsymbol{x}_{in}) \right. \\ &\quad \left. - \frac{1}{N_{C_1}} \sum_{j=1, j\in C_1}^{N_{C_1}} \left(\sum_n \alpha_{jn} \Phi(\boldsymbol{x}_{jn}) \right) \right\|^2 \\ &= \left\| \sum_n \alpha_{in} \Phi(\boldsymbol{x}_{in}) \right\|^2 \\ &\quad + \left\| \frac{1}{N_{C_1}} \sum_{j=1, j\in C_1}^{N_{C_1}} \left(\sum_n \alpha_{jn} \Phi(\boldsymbol{x}_{jn}) \right) \right\|^2 \\ &\quad - \frac{2}{N_{C_1}} \left(\sum_n \alpha_{in} \Phi(\boldsymbol{x}_{in}) \right) \\ &\quad \times \left(\sum_{j=1, j\in C_1}^{N_{C_1}} \left(\sum_n \alpha_{jn} \Phi(\boldsymbol{x}_{jn}) \right) \right). \end{aligned} \tag{6}$$

This is like (2) for one cluster and one class i in higher-order transformed space. As seen, all terms are inner product terms, easily evaluated by kernel methods, $\Phi(\boldsymbol{x}_i)^{\mathrm{T}} \Phi(\boldsymbol{x}_j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma^2)$; thus, (6) is easy to compute. Note that all summations in (6) are only over support vectors. In each iteration of our WSV k-means clustering process, we compare $d_1 = \|\boldsymbol{h}_i - \Phi(\boldsymbol{m}_1)\|$ and $d_2 = \|\boldsymbol{h}_i - \Phi(\boldsymbol{m}_2)\|$ for all of the $\boldsymbol{h}_i$ for all classes present; if $d_1 < d_2$ for a specific $\boldsymbol{h}_i$, that $\boldsymbol{h}_i$ is assigned to cluster C_1 (i.e. class i is placed in macro-class 1) and vice versa. After all solution vectors $\boldsymbol{h}_i$ are grouped into one of the two clusters C_1 and C_2, new cluster means $\Phi(\boldsymbol{m}_1)$ and $\Phi(\boldsymbol{m}_2)$ are updated, and the clustering process is repeated until the total distance between each $\boldsymbol{h}_i$ and its cluster mean is minimized (as in the standard k-means clustering algorithm). *This method is a major new method in visualizing and clustering high-dimensional support vector data.*

Thus, for a node with M classes to be separated, we first obtain M SVRM solution vectors $\boldsymbol{h}_i$ for the M classes in transformed space. Next, we apply our new k-means clustering algorithm ($k = 2$) to these M vectors (recall that each $\boldsymbol{h}_i$ is a weighted sum of all support vectors for class i), and we iterate to divide these M vectors into two sets. Therefore, two clusters (macro-classes) are determined at each node, for which (1) in high dimensional space is optimized for both clusters. Note that the distances in (6) are calculated using a weighted sum of support vectors; we thus refer to this new algorithm as *weighted support vector k-means clustering*. Since no validation set data is needed in this method (our earlier (Casasent & Wang, 2005) balanced binary design method needs validation set data), this design is also much faster. Once the macro-class pair for each node in the hierarchy has been selected, the design of the binary hierarchical classifier is complete. We then form an SVRDM classifier (Section 3) for the macro-class pair chosen at each node using training set samples, and we use them as the binary classifiers at each node in the hierarchy.

3. Support vector representation and discrimination machine (SVRDM)

3.1. SVRDM algorithm

The SVRDM is a version of the SVM that provides good discrimination between the true classes (as the SVM does) and better rejection for false classes (Yuan & Casasent, 2003) (than the SVM can). In the SVM, to classify an input $\boldsymbol{x}$ into one of two classes, we compute the VIP (vector inner product) evaluation function output of the transformed test input $\Phi(\boldsymbol{x})$ and the nonlinear filter function $\boldsymbol{h}$ as $f(\boldsymbol{x}) = \boldsymbol{h}^{\mathrm{T}} \Phi(\boldsymbol{x})$. In the standard SVM, $\boldsymbol{h}$ is computed from the training set and solves the following quadratic programming problem

$$\text{Min } \|\boldsymbol{h}\|^2 / 2, \text{ s.t. } \begin{cases} \boldsymbol{h}^{\mathrm{T}} \Phi(\boldsymbol{x}_i) \geq +1, & i = 1, 2, \ldots, N_1 \\ \boldsymbol{h}^{\mathrm{T}} \Phi(\boldsymbol{x}_j) \leq -1, & j = 1, 2, \ldots, N_2, \end{cases} \tag{7}$$

where N_1 and N_2 are the number of training set samples in each class. For our hierarchical classifier, the two classes to be separated at each node are the associated pair of macro-classes. Since the standard SVM cannot reject false class inputs well (Tax & Duin, 1999), we extend the SVM to our new SVRDM to achieve better rejection performance. In our hierarchical SVRDM classification structure, each node handles classification of two classes (macro-classes). The two

solution vectors $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ for classes (macro-classes) 1 and 2 at a given node are chosen to satisfy

$$\begin{aligned}
&\text{Min } \|h_1\|^2/2\\
&h_1^{\mathrm{T}}\Phi(x_{1i}) \geq T, i = 1, 2, \ldots, N_1\\
&h_1^{\mathrm{T}}\Phi(x_{2j}) \leq p, j = 1, 2, \ldots, N_2,\\
&\text{Min } \|h_2\|^2/2\\
&h_2^{\mathrm{T}}\Phi(x_{2j}) \geq T, j = 1, 2, \ldots, N_2\\
&h_2^{\mathrm{T}}\Phi(x_{1i}) \leq p, i = 1, 2, \ldots, N_1\,.
\end{aligned} \tag{8}$$

In (8), there are N_1 and N_2 samples in each class (macro-class). We note that the second class output is specified to be $\leq p$ (and not $-T$ or -1, as in the standard SVM). If $p = -1$, then (8) describes the standard SVM. We use $p \neq -1$; this improves rejection (as we have shown Yuan and Casasent (2003)). Typically, we choose p in the range $[-1, 0.6]$. We use $T = 1$ in training and vary it in testing. In the presence of outliers (training class errors), slack variables ξ are used in both $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$. Thus, the final $\boldsymbol{h}_1$ in (8) satisfies

$$\begin{aligned}
&\text{Min}\left\{\|h_1\|^2/2 + C\left(\sum \xi_{1i} + \sum \xi_{2j}\right)\right\},\\
&\quad \xi_{1i} \geq 0,\ \xi_{2j} \geq 0\\
&h_1^{\mathrm{T}}\Phi(x_{1i}) \geq T - \xi_{1i}, \quad i = 1, 2, \ldots, N_1\\
&h_1^{\mathrm{T}}\Phi(x_{2j}) \leq p + \xi_{2j}, \quad j = 1, 2, \ldots, N_2\,.
\end{aligned} \tag{9}$$

In (9), C is the weight of the penalty term for the slack variables, which is the tradeoff between the amount of errors and the separation between the two classes (Cortes & Vapnik, 1995). We use $C = 20$. The final version of $\boldsymbol{h}_2$ is similar.

At each node in the hierarchical design, we evaluate the associated two VIPs of the transformed input $\Phi(\boldsymbol{x})$ and the $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ (at that node). The VIP with the largest output ($\geq$ threshold T) denotes the macro-class decision made at that node. If neither VIP gives an output $\geq T$, the test input is rejected as a false class. As in an SVM, only the VIP of the transformed test input $\Phi(\boldsymbol{x})$ and the nonlinear filter function $\boldsymbol{h}$ is needed; the explicit form of the transformed Φ is not necessary (see Section 2.2.1). We use the Gaussian kernel function for reasons discussed in Section 3.2.

3.2. Basis for SVRDM good rejection & SVRDM parameter choices

In SVM-type classifiers, $\boldsymbol{h}^{\mathrm{T}}\Phi(\boldsymbol{x})$ is evaluated using kernel functions $K(\boldsymbol{x}_i, \boldsymbol{x})$, where the $\boldsymbol{x}_i$ are the support vectors and $\boldsymbol{x}$ is the test input. Several possible kernel choices exist, such as linear, polynomial, Gaussian, etc. However, to address both classification and rejection problems, we want the kernel function to give large output values for true class inputs and low values for false classes (to be rejected). The Gaussian kernel $K(\boldsymbol{x}_i, \boldsymbol{x}) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}\|^2/2\sigma^2)$ approaches zero as $\|\boldsymbol{x}_i - \boldsymbol{x}\|$ increases, i.e. when the test input $\boldsymbol{x}$ is far from the support vector $\boldsymbol{x}_i$. This is likely to be the case for false class inputs; their evaluation function output will thus be below the threshold T and they will be rejected. If a polynomial kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}) = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}_i + 1)^d$ is used, the VIP $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}_i$ can be very large when $\boldsymbol{x}$ is far from $\boldsymbol{x}_i$ and lies in the same direction as $\boldsymbol{x}_i$.

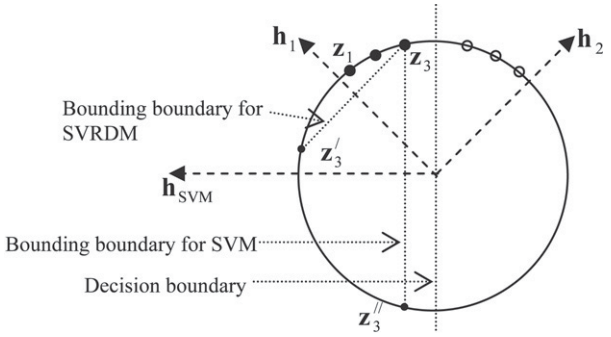

Fig. 2. Conceptual SVRDM ($\boldsymbol{h}_1$) and SVM ($\boldsymbol{h}_{\mathrm{SVM}}$) solutions.

Therefore, for a polynomial kernel, the evaluation function output may still be large ($>T$) for false class inputs, and thus this kernel is not good when rejection is of concern. Similar remarks can also be used to show that linear, sigmoid, etc., kernels are not good for rejection.

Next, by illustration, we show why our SVRDM with a Gaussian kernel is better than a standard SVM at rejection. For the Gaussian kernel, all data lie on the unit sphere in transformed space. Fig. 2 conceptually shows the solution vectors $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ (for the SVRDM) and $\boldsymbol{h}_{\mathrm{SVM}}$ (for the SVM) in the transformed space. For simplicity, training data points in Fig. 2 (filled in and open circles) denote the extremes of the training set and the average training set vector for each class. The SVM solution vector $\boldsymbol{h}_{\mathrm{SVM}}$ is normal to the decision boundary, which is a vertical line through the center of the circle. For our SVRDM with $p = 0.6$, the vector $\boldsymbol{h}_1$ shown satisfies (8). The $\boldsymbol{h}_2$ solution vector for class 2 is similar to $\boldsymbol{h}_1$ but it lies in the first quadrant, symmetric about the vertical for this example. The lengths of the three vector solutions shown are proportional to their norms (energy). We note that the length, or equivalently, the energy of $\boldsymbol{h}_{\mathrm{SVM}}$ is larger than that of $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$. This is expected, because a larger $p > -1$ is used in (8) and (9) for the SVRDM (*this is a looser constraint in the quadratic programming problem, and thus a better minimum point can be found*). The decision region of acceptance of true class data is proportional to the energy of the solution vector. Thus, the SVM has a larger acceptance region and hence poorer rejection than the SVRDM due to the lower p value (looser constraint) in the SVRDM. In Fig. 2, the range of transformed inputs that will be accepted as class 1 for the SVRDM is described by the arc z_3'–z_3. This range is much less than the z_3'' to z_3 range for the SVM. Thus, we expect a lower false alarm rate P_{FA} and hence better rejection for our SVRDM (with $p = 0.6$) vs. the SVM (with $p = -1$). If Fig. 2 were a hyper-sphere, the bounding boundary (within which $\Phi(\boldsymbol{x})$ satisfies $\boldsymbol{h}^{\mathrm{T}}\Phi(\boldsymbol{x}) \geq 1$) for the SVM would be a hyper-circle on the surface of the hyper-sphere as shown in 2-D in Fig. 2.

To train an SVRDM, we must choose values for several parameters. If σ in the Gaussian kernel function is chosen too small, the decision region for the class samples will be too tight, which is analogous to overfitting and results in poor generalization. As σ decreases, more training samples become support vectors. We developed a new automated method to select σ (Yuan & Casasent, 2005). *Our σ selection method and*

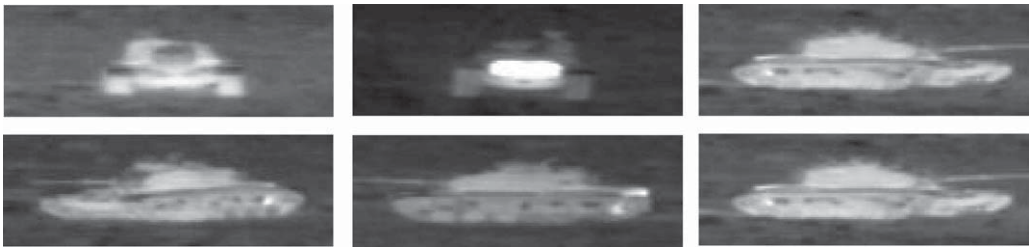

Fig. 3. The M60 tank at (counterclockwise) 0°, 45°, 90°, 270°, 315°, and 180° aspect views (starting from the upper left image).

our use of $p > -1$ are the two major differences in our SVRDM and the standard SVM. Note that larger T in testing (and a larger p in training) give smaller decision regions and thus better P_{FA}. In this paper, we use $p = 0.6$ and $C = 20$ (which is not critical). We varied T to achieve different operating points in our results in Section 6.

4. Time complexity comparison

We now show that a binary hierarchical classification structure such as ours using SVM type classifiers is more efficient than the standard one-vs-rest and one-vs-one methods by comparing the required training and testing times.

4.1. Training

Empirically, the training time for an SVM has been observed (Platt, 1999) to be $T \propto N_T^{\gamma}$, where N_T is the total training set size and γ is typically assumed to be 2. For a C-class problem with total training set size N_T, we assume that each class has the same number of training samples N_T/C. To train C different one-vs-rest SVM classifiers require CN_T^2. For the one-vs-one method, we need to train $C(C-1)/2$ different SVMs, each with a training set size $2N_T/C$. Therefore, the training time required is $(C(C-1)/2)(2N_T/C)^2 \approx 2N_T^2$. With the same assumptions, for a balanced binary hierarchical structure of SVMs, we need to train the SVM at the top node with the full training set size N_T, the two SVMs at the second level each has a training set size of $N_T/2$, the four SVMs at the third level each has a training set size of $N_T/4$, etc.; at the last $(\log_2 C)$ level, there are $C/2$ different SVMs. Therefore, the training time required is $(N_T^2 + 2(N_T/2)^2 + 4(N_T/4)^2 + \cdots) \approx 2N_T^2$. Note that the time to train a binary hierarchical structure with a total of $C-1$ SVMs is the same as that of the one-vs-one method.

4.2. Testing

We now compare the time complexity for testing each method. From (4), we need to evaluate $\boldsymbol{h}^{\mathrm{T}}\Phi(\boldsymbol{x})$ for each SVM. With a Gaussian kernel, $\boldsymbol{h}^{\mathrm{T}}\Phi(\boldsymbol{x})$ is

$$\sum_{i}^{N_S} \Phi(\boldsymbol{x}_i)^{\mathrm{T}}\, \Phi(\boldsymbol{x}) = \sum_{i}^{N_S} \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}\|^2 / 2\sigma^2\right), \tag{10}$$

where N_S is the number of support vectors. Therefore, instead of performing each inner product $\Phi(\boldsymbol{x}_i)^{\mathrm{T}}\Phi(\boldsymbol{x})$, we only need to evaluate each of the Gaussian kernel functions. For simplicity, we assume that a fraction α $(0 < \alpha < 1)$ of the training set data are support vectors, i.e. $N_S = \alpha N_T$. In the one-vs-rest method, the number of computations we need for an SVM is approximately αN_T; therefore, the total number of computations to classify a test input is approximately $\alpha C N_T$. For the one-vs-one method, we require about $\alpha x C(C-1)/2x(2N_T/C) \approx \alpha C N_T$ calculations to obtain the outputs from all $C(C-1)/2$ SVMs. For the binary hierarchical method, the test time required is $\alpha(N_T + N_T/2 + N_T/4 + \cdots) \approx 2\alpha N_T$, which is less than the standard methods by a large factor of $C/2$ (recall that C is large). Therefore, we conclude that the binary hierarchical classification structure is superior to other standard multi-class SVM algorithms in terms of training and testing time. For the SVRDM, an extra factor of 2 is necessary because two evaluation functions must be evaluated. In Section 6, we compare the performance and the time complexity of these three classification approaches.

5. Database description (Comanche)

The Comanche database is a real infrared (IR) database. The eight object classes are: 2S1, BMP, HMMWV, M1, M113, M3, M60 and M730 (referred to as classes 1–8). The images used are at a 2 km range with 72 aspect views of each object in 2 thermal states available. Each target chip is 151 × 51 pixels; Fig. 3 shows the M60 tank images at six different aspect views.

Fig. 4(a) and (b) show the eight targets (classes 1 and 2 are in the top row, etc.) in the two different thermal states (classes 4 and 7 are tanks; classes 1, 4, 6 and 7 are tank-like; class 5 is an APC, etc.) The backgrounds are also different. The objects in the Comanche database are bimodal and have low resolution (*since only detection, not classification, was considered in the Comanche program*). For each thermal state, we use 16 aspect views in training and 56 in testing; thus, the test set data differs from training set data by up to 25° in aspect. There are thus $16 \times 2 = 32$ training samples and $56 \times 2 = 112$ test images per target class.

To evaluate the rejection ability of our SVRDMs, we consider rejecting 21 different aspect views of the M2 (tank), M35 (truck), and M163 (APC) targets, each with four different thermal states, from a synthetic IR (TRIM-2) database. The TRIM-2 database has twelve classes (eight vehicles and four aircraft).The three TRIM-2 targets to be rejected are not present in the Comanche database; there are thus $3 \times 4 \times 21 = 252$ false objects to be rejected. *Note that none of these false object images are used in the training set; this is realistic and makes the rejection problem very difficult.*

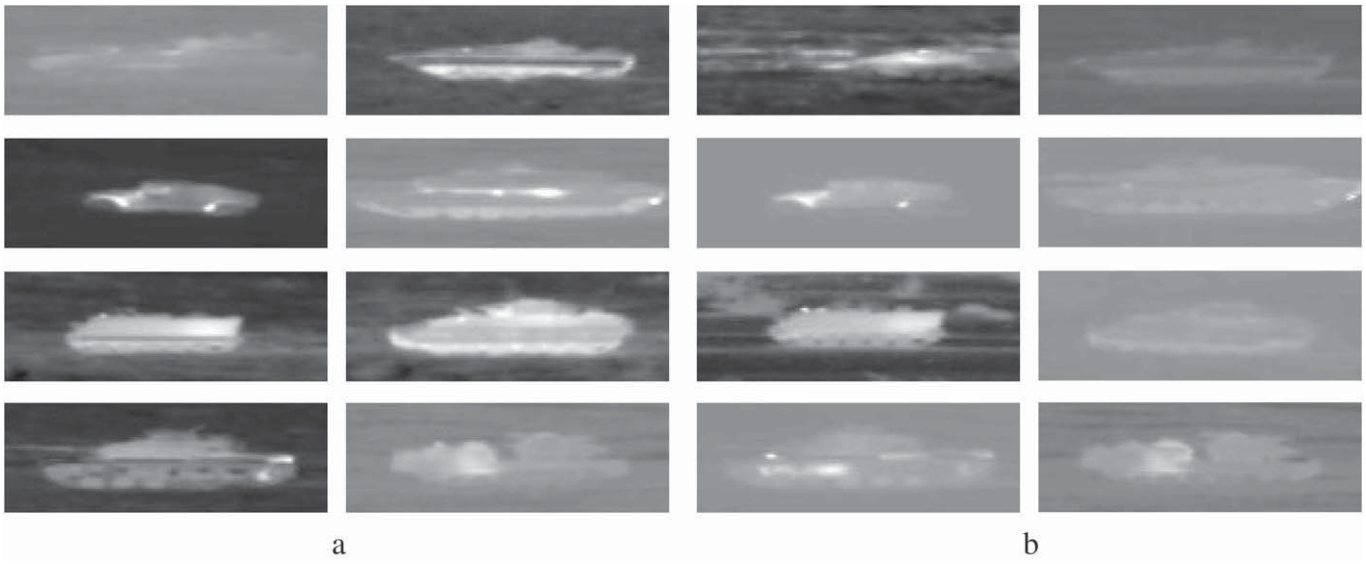

Fig. 4. (a) The first and (b) the second thermal states of broadside images of the eight classes to be recognized in the Comanche database.

6. Feature selection

We use magnitude Fourier Transform (|FT|) features in training and testing, *because they are shift-invariant* (the features remain the same even if the target is shifted in the region of interest). This property is especially needed in ATR tracking and classification applications, since the center of the target in the scene cannot be exactly determined. We have confirmed (Casasent & Wang, 2005) that if we used *pixel-based features*, then if the target is moved by only 2–3 pixels, the SVRDM classifier (or any pixel-based classifier) will not give high performance. Since standard wavelets are not shift-invariant, they were not considered in our work. Since the |FT| plane is symmetric, another advantage is that we need only half of the |FT| features. We also ignore higher |FT| components; this further reduces the number of features used. For an $m \times n$ pixel target chip, we use the lower $m/4 \times n/2$ (vertical $\times$ horizontal) $= (m \times n)/8$|FT| components in the first and the second quadrants as our feature space. This use of $75 \times 13 = 975$ features saves a factor of 8 in computation time. We remove the DC point in the |FT| plane (this is a form of image normalization that is of use with targets with thermal variations), since the DC value is very dominant (about 1 or 2 orders larger than other |FT| components). Image preprocessing techniques to reduce background noise, which is not the major concern of this paper, can also be used to improve performance. For future training images, we will use target images in a constant background (i.e. without noise), and test images will be targets present in different types of real background; this will indicate the generalization of our classifier in realistic applications.

7. Experimental results

In prior work, we designed a binary hierarchical classification structure, using our old balanced design method, to classify eight vehicle classes and to reject four aircraft classes (all data were in the synthetic IR TRIM-2 database) and obtained only one mis-classification error. Using our new WSV k-means clustering design, we obtained perfect classification $P_C = 100\%$ and false alarm $P_{FA} = 0\%$ values. In the present paper, we consider the real IR (Comanche) data; its thermal state variations are much more severe than those in TRIM-2 data (in which each target was approximately all hot or cold); thus this present problem is much more difficult. Fig. 5 shows the hierarchical structure produced by our new design. The numbers in each box denote the classes in each macro-class at a given node. For example, at the top node, our new clustering algorithm separates the original eight classes into two macro-classes; classes 1, 2, 5, 6, and 7 are in macro-class 1, and classes 3, 4, and 8 are in macro-class 2. We observe that the two tanks (classes 4 and 7) are not in the same macro-classes in the hierarchy. Without an automated hierarchical design method, one would use intuition and place all tanks in the same macro-class. Therefore, *a practical and automated hierarchy design method (like ours) is very important*. If a balanced binary hierarchical design is used, classes 1, 3, 4, 7 and classes 2, 5, 6, 8 were the two macro-classes at the top node; if the design method proposed in Wang and Casasent (2006) is used (k-means based support vector clustering in the

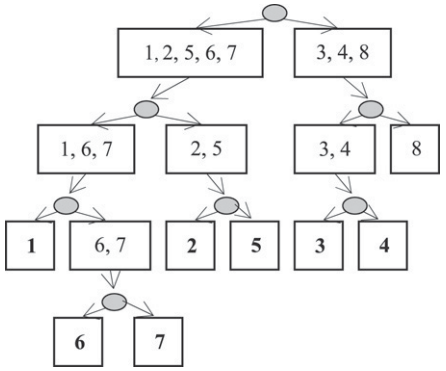

Fig. 5. Binary hierarchical classification structure (8 classes) designed by our WSV k-means clustering algorithm for the Comanche database.

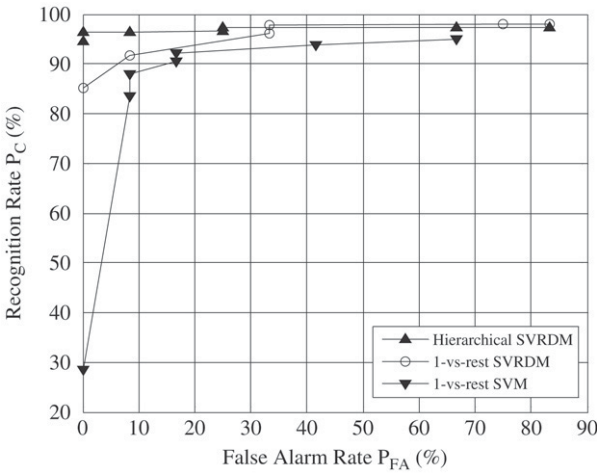

Fig. 6. ROC curves for different classification strategies.

Table 1
Comparisons of test results using different classification strategies: (a) our hierarchical SVRDM classification structure, (b) the 1-vs-rest method with SVRDMs, and (c) with SVM classifiers

	SVRDM ($p = 0.6$)		SVM ($p = -1$)
	(a) Our method (%)	(b) 1-vs-rest (%)	(c) 1-vs-rest (%)
P_C@$P_{FA} = 8.3\%$	96.4	91.7	87.9
P_{FA}@$P_C 95\%$	0	26.7	66.7
EER (%)	3.6	8.3	11.2

original data space), classes 1, 2, 5, 6 and 3, 4, 7, 8 were the two macro-classes at the top node. These hierarchical structures are both different from the one using the new proposed method in this paper.

Fig. 6 shows the ROC curves, P_C vs. P_{FA}, for three different classification strategies: our hierarchical SVRDM classifier and standard one-vs-rest classifiers (using SVRDMs and SVMs, respectively). Each point on the ROC curve corresponds to a given threshold T choice in the SVM/SVRDM evaluation function. From Fig. 6, we see that our hierarchical SVRDM classifier outperforms the two standard methods, since its ROC curve is above and to the left of them, while the one-vs-rest SVM performs worst.

Next, we compare their classification performances at the same $P_{FA} = 8.3\%$ point. From the first row in Table 1, we see that our method successfully recognized 96.4% of the true class inputs, the others achieved $P_C = 91.7\%$ and 87.9%. To compare the rejection ability of these classifiers, we consider the $P_C = 95\%$ point and compare P_{FA} results. The second row of Table 1 indicates that our hierarchical SVRDM classifier has perfect $P_{FA} = 0\%$ and gives the best rejection performance, while the other classifiers give P_{FA} of 26.7% and 66.7%. As noted earlier, the standard SVM has poor rejection ability, and, in general, we expect worse performance using the 1-vs-rest method, because each classifier has a harder problem to solve than in the hierarchical method. When the number of classes is larger, we expect the differences between the hierarchical and one-vs-rest SVRDM cases to be even larger. As shown in the last row in Table 1, our hierarchical SVRDM classifier has the lowest equal error rate ($P_C = P_{FA}$) EER = 3.6%, the two standard methods only have EER = 8.3% and 11.2%. We did not consider a hierarchical classifier using SVMs, since the standard SVM has poor rejection, as indicated in Table 1.

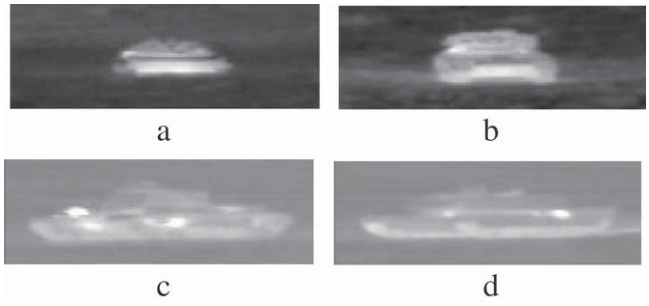

Fig. 7. (a) The misclassified BMP (at 355°) is recognized as an M3, (b) an M3 (at 355°) looks similar to (a), (c) the misclassified M60 (at 45°) is recognized as an M1, (d) an M1 (at 30°) looks similar to (c).

Table 2
Comparisons of time complexity for different classification strategies

	SVRDM ($p = 0.6$)		SVM ($p = -1$)
	(a) Our method	(b) 1-vs-rest	(c) 1-vs-rest
Training	$238N^2$	$1024N^2$	$512N^2$
Testing	$30\alpha N \approx 13N$	$64\alpha N \approx 20N$	$64\alpha N \approx 24N$
(α value)	(∼0.45)	(∼0.31)	(∼0.37)

N is the training set size for one class, and α is the fraction of the training set data used as support vectors.

Fig. 7 shows some mis-classification examples for our hierarchical SVRDM classifier at $P_{FA} = 8.3\%$: the BMP at a 355° aspect view (Fig. 7(a)) was recognized as an M3 (Fig. 7(b) shows an M3 at the same aspect view), the M60 tank at a 45° aspect view (Fig. 7(c)) was recognized as an M1 tank (Fig. 7(d) shows an M1 tank at a 30° aspect view). It is not easy to determine exactly why these errors occur; however, most Comanche target images have poor resolution (since detection, not classification, was considered in the original Comanche program), and thus it is difficult to classify these target images perfectly.

Table 2 lists the training times for all classifiers and the testing times for classifying one input image. Here, α is the fraction of the training set data considered as support vectors, and N is the training set size for a single class ($N_T = 8N$, see Section 4). As seen, to train $C - 1 = 7$ SVRDMs in our hierarchical structure, the training time is $238N^2$ or 24 min on our hardware. To train the $C = 8$ SVRDMs using the one-vs-rest method takes much more time: $1024N^2 > 238N^2$ or 125 min. This training time is off-line and is not of major concern. The testing time for our hierarchical SVRDM classifier depends on the hierarchical structure designed and the class of the test input; as shown in Fig. 5, we only need to use two SVRDMs (the best case) to classify test inputs from class 8 (the shortest path in Fig. 5) and the testing time for this class is $8\alpha N$ (the time for the SVRDM with 8 classes at the top node) + $3\alpha N$ (the time for the SVRDM with 3 classes at the second level) = $11\alpha N$; however, a path of 4 SVRDMs (worst cases) is needed

to classify test inputs in classes 6 or 7, and thus their testing time is $18\alpha N$ ($8\alpha N + 5\alpha N + 3\alpha N + 2\alpha N$, from the top node to the bottom). We note that the training time ($238N^2$ and $1024N^2$) for SVRDM-based approaches, and the average test time ($30\alpha N$) for the hierarchical SVRDM classifier listed in Table 2, include the extra factor of 2 noted in Section 4.2. The test time for the 1-vs-rest SVRDM and SVM classifiers requires $64\alpha N$ (8 classifiers $\times$ $8\alpha N$ support vectors) or about 0.2 s. All test times are small, but our hierarchical SVRDM classifier requires the least. These computations were made in Matlab on a P4 1.8 GHz PC with 768 MB RAM.

8. Conclusion and future work

A novel WSV K-means clustering method was proposed to design the binary hierarchical structure. This performs the design in transformed high dimensional space; this is very new and important because, unlike prior work on hierarchical design, our design method and our SVRDM binary classifiers (applied at each node in the hierarchy) both operate in high dimensional space. We also explained why a Gaussian kernel gives the best rejection of any SVM. Our preliminary results have shown excellent classification and rejection results on a real eight-class IR database. Compared to the standard one-vs-rest classifiers, our hierarchical classifier gives better P_C and P_{FA}; this occurs since the classifiers in the hierarchy are simpler than the one-vs-rest classifiers. Our classifier is also seen to be more computationally efficient. In future work, we can test our method on larger class problems (e.g. face, fingerprint, handwritten character, etc., recognition), and in such cases we expect larger performance differences between our method and standard methods.

References

Anand, R., Mehrotra, K., Mohan, C. K., & Ranka, S. (1995). Efficient classification for multiclass problems using modular neural networks. *IEEE Transactions on Neural Networks*, *6*, 117–124.

Casasent, D., & Wang, Y. -C. (2005). A hierarchical classifier using new support vector machine for automatic target recognition. *Neural Networks*, *18*(2), 541–548.

Chen, Y.Q., Zhou, X.S., & Huang, T.S. (2001). One-class SVM for learning in image retrieval. In *Proceedings of the international conference on image processing* (pp. 34–37).

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning*, *20*, 1–25.

Duda, R., Hart, P., & Stork, D. (2000). *Pattern classification*. New York: Wiley-Interscience.

Fukunaga, K. (1990). *Introduction to statistical pattern recognition* (2nd ed.). Boston: Academic Press.

Hastie, T., & Tibshirani, R. (1998). Classification by pairwise coupling. *Advances in neural information processing systems*, *10*, 507–513.

Kumar, S., Ghosh, J., & Crawford, M. M. (2002). Hierarchical fusion of multiple classifiers for hyperspectral data analysis. *Pattern Analysis and Applications*, *5*(2), 210–220. Special issue on fusion of multiple classifiers.

Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. *Advance in Neural Information Processing Systems*, *10*, 336–342.

Platt, J. C., Cristianini, N., & Shawe-Taylor, J. (1999). Large margin DAGs for multiclass classification. *Advances in Neural Information Processing Systems*, *12*, 547–553.

Shaik, J., & Yeasin, M. (2006). A progressive framework for two-way clustering using adaptive subspace iteration for functionally classifying genes. In *Proceedings of the international joint conf. on neural networks* (pp. 2980–2985).

Tax, D., & Duin, R. (1999). Data domain description using support vectors. In *Proceedings of european symposium on artificial neural networks* (pp. 251–256).

Vural, V., & Dy, J.G. (2004). A hierarchical method for multi-class support vector machines. In *Proceedings of the international conference on machine learning* (pp. 105–112).

Wang, Y.-C.F., & Casasent, D. (2006). Hierarchical k-means clustering using new support vector machines for multi-class classification. In *Proceedings of the international joint conf. on neural networks* (pp. 3457–3464).

Wang, Y.-C.F., & Casasent, D. (2007). New weighted support vector k-means clustering for hierarchical multi-class classification. In *Proceedings of the international joint conf. on neural networks*.

Yuan, C., & Casasent, D. (2003). Support vector machines for class representation and discrimination. In *Proceedings of the international joint conference on neural networks* (pp. 1611–1616).

Yuan, C., & Casasent, D. (2005). Face recognition and verification with pose and illumination variations and imposter rejection. *Proceedings of the SPIE*, *5779*(29), 242–255.

Yu-Chiang Frank Wang received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, in 2001. From 2001 to 2002, he worked in the Division of Medical Engineering Research at the National Health Research Institutes, Taiwan, as a research assistant, where he was working on the design and development of ultrasound CT imaging systems. He received his M.S. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, Pennsylvania, in 2004. He is currently a Ph.D. student in Carnegie Mellon University, and his research interests are computer vision and pattern recognition.

David Casasent is a professor at Carnegie Mellon University, Pittsburgh, Pennsylvania, in the Department of Electrical and Computer Engineering, where he holds the George Westinghouse Chair. He is a fellow of the IEEE, OSA, and SPIE and has received various best paper awards and other honors. He is past president of SPIE and the International Neural Network Society. He is the author of two books, editor of one text, editor of 70 journals and conference volumes, and contributor to chapters in 20 books and more than 700 technical publications, on various aspects of optical data processing, image pattern recognition, and product inspection.