A vector-valued support vector machine model for multiclass problem

Ran Wang, Sam Kwong, Degang Chen, Jingjing Cao

A vector-valued support vector machine (SVM) model is proposed for multiclass problems. The basic idea is to separate 2^c classes by SVM hyperplanes in the feature space induced by certain kernels, where c is a finite positive integer. We start from a 2^c-class problem, and extend it to any-class problem by applying a hierarchical decomposition procedure. Compared with the existing SVM-based multiclass methods, the VVD model has two advantages. First, it reduces the computational complexity by using a small number of classifiers. Second, the feature space partition induced by the hyperplanes effectively eliminates the unclassifiable regions (URs) that may affect the classification performance of the algorithm. Experimental comparisons with several state-of-the-art multiclass methods demonstrate that VVD maintains a comparable testing accuracy, while it improves the classification efficiency with less classifiers, a smaller number of support vectors (SVs), and shorter testing time.

1. Introduction

Support vector machines (SVMs) have attracted significant interest during recent years and have exhibited outstanding performances on various learning domains, such as data mining, syndrome recognition, hidden Markov modeling, protein prediction, and financial time series forecasting. Traditional SVMs are typically designed for binary classification problems. Extending them to multiclass classification remains a hot topic.

Vapnik first proposed the one-against-all (OAA) method which decomposes an A-class problem into A binary problems, where A is a finite positive integer and A ≥ 3. This method has been very popular, but suffers from high training expenses and the unclassifiable region (UR) problem. Inoue et al. proposed the membership function to identify the unclassifiable patterns of OAA, and Ryan et al. presented the comparable performance of OAA against other approaches, but the high training expenses remain as a problem yet to be solved.

Friedman later raised the one-against-one (OAO) approach which solves an A-class problem via A(A - 1)/2 binary classifiers by adopting a pairwise method. OAO has a faster training speed than OAA, but it also suffers from the UR problem.
that may decrease the classification ability of the algorithm. Platt et al. [27] and Kijserikul et al. [17] respectively proposed Decision Directed Acyclic Graph (DDAG) and adaptive-directed-acyclic-graph (ADAG) to handle the URs by building up some graph-based architectures. Later, three variants of OAO were proposed, i.e., membership function [36], nesting-OAO algorithm [23], and vector projection method [20]. Although all these approaches can handle the URs to a certain extent, they also increase the model complexity. Besides, the numerous classifiers may lead to the overfitting problem since they contain much overlapped information.

Different from OAA and OAO, the Decision Tree (DT) based method [3,7,25,31,34] builds up some class decomposition rules via a binary tree structure. For an \(A\)-class problem, the binary tree usually contains \(A-1\) tree nodes, and each node is a classifier that makes binary decisions among the classes. The performance always relies on different DT structures.

It was concluded in two previous studies [13,42] that none of the above-mentioned methods can significantly outperform the others in term of testing accuracy. However, they suffer from two problems that may affect the classification ability or efficiency, including (1) intensive computation with numerous classifiers and (2) URs. In this paper, a new model named Multiclass SVMs with Vector-Valued Decision (M-SVMs-VVD), or VVD in short, is proposed. We developed this model with two motivations:

- To solve a multiclass problem with the least number of classifiers, and at the same time maintain a comparable accuracy.
- To eliminate the URs during the training process, rather than handle them with additional efforts in the testing process.

With the least number of classifiers, the support vectors (SVs) are cut down, thus the computational complexity for both training and testing can be reduced. In addition, by partitioning the feature space in a more effective manner, URs are fully eliminated and the testing efficiency can be improved.

Given that \(a\) is a finite positive integer, the basic idea is to separate \(2^a\) classes by \(a\) SVM hyperplanes in the feature space induced by certain kernels, thus to classify examples via \(a\)-dimensional boolean-valued vectors. Therefore, the VVD model was first designed for \(2^a\)-class problem. Then, by building up a hierarchical class decomposition structure, this model was further extended to any-class problem.

The rest of this paper is organized as follows: in Section 2, the standard formulation of SVMs and the several most widely used multiclass approaches are reviewed. In Section 3, some important theories, as the basis of the proposed method, are introduced. In Section 4, the VVD model is built up, and some related analyses are then described in detail. In Section 5, two important issues with regard to the VVD model, i.e., the effects of unbalanced data and the labeling order of classes, are discussed. In Section 6, experimental comparisons with several state-of-the-art approaches are conducted to assess the performances of the proposed VVD model. Finally, conclusions are given in Section 7.

2. SVMs and Multiclass Classification Approaches

This section reviews the basic formulation of SVMs and several most widely used multiclass approaches.

2.1. Formulation of SVMs

SVMs are famous classification techniques based on statistical learning theory [37]. Given a binary problem on the training set \(\{(x_i, y_i)\}_{i=1}^n \in \mathbb{R} \times \{\pm 1\}\), the key idea is to produce an optimal separating hyperplane that can maximize the margin between the two classes. The hyperplane is defined as \(w^T x + b = 0\), where \(w\) is a \(l\)-dimensional vector, \(x\) represents the argument, and \(b\) is a constant value. Thus, the problem for the linearly separable case could be formulated as (1).

\[
\begin{align*}
\min_{w,b} & \quad \frac{1}{2}w^Tw \\
\text{s.t.} & \quad y_i(w^T x_i + b) \geq 1 \\
& \quad i = 1, \ldots, n
\end{align*}
\]

(1)

With Lagrangian method, (1) could be transferred into a Quadratic Programming (QP) problem to derive the optimal hyperplane.

In case the data are nonlinearly separable in the input space, there are two approaches to handle the situation. The first approach is called soft-margin SVMs which transfers the problem into (2).

\[
\begin{align*}
\min_{w,b,\xi} & \quad \frac{1}{2}w^Tw + C \sum_{i=1}^n \xi_i \\
\text{s.t.} & \quad y_i(w^T x_i + b) \geq 1 - \xi_i \\
& \quad \xi_i \geq 0, \quad i = 1, \ldots, n
\end{align*}
\]

(2)

where \(\xi_i\) is the slack variable introduced to \(x_i\), and \(C\) is a trade-off constant. The other approach is to map the data from the input space into a higher dimensional feature space via a mapping function \(\phi: \mathbb{R} \rightarrow \phi(\mathbb{R})\), which makes them linearly separable. Since the solution in feature space involves only inner products of the mapped points, one can obtain the optimal hyper-
plane by kernel trick [8], which expresses the inner product of feature space as a kernel function \( k: \langle \phi(x), \phi(x') \rangle = k(x, x') \).

Finally, the decision function is achieved as (3),

\[
h(x) = \sum_{i=1}^{n} y_i z_i k(x, x_i) + b
\]  

(3)

and \( f(x) = \text{sign}(h(x)) \) is determined as the classification output, where \( z_i \) is the lagrange multiplier of \( x_i \).

Improved works of SVMs have been later proposed in some studies [21,26,33]. However, it is noteworthy that these techniques are specially designed for binary classification problems and researchers continue making some efforts for extending them to multiclass classification.

### 2.2. Multiclass Classification Approaches

#### 2.2.1. One-against-all (OAA)

The OAA approach constructs \( A \) binary classifiers, i.e., \( f_1, \ldots, f_A \), where \( f_i(x) = \text{sign}(h_i(x)) \), \( i = 1, \ldots, A \), separates class \( i \) from the rest \( A-1 \) classes. For the testing example \( x' \in \mathbb{R}^l \), if it satisfies \( f_{i_1}(x') = 1 \) and \( f_{i_2}(x') = -1 \), where \( i_1 = 1, \ldots, A \) and \( j \in [1, \ldots, A] \), then the output class label is determined as \( j \). Otherwise, it falls into the URs. To handle this situation, the membership function proposed in a previous study [14] could be adopted. In fact, it is not necessary to compute the membership function, since the example’s output class could be directly determined by its maximum decision value among the \( A \) classifiers, i.e., \( \text{output} = \arg\max_{i=1, \ldots, A} (h_i(x')) \).

OAA has a simple implementation procedure and a comparatively smaller number of classifiers, but it needs to solve \( A \) QP problems that involve all training examples, thus the training cost is expensive.

#### 2.2.2. One-against-one (OAO)

The OAO approach constructs \( A(A-1)/2 \) binary classifiers and each classifier is trained with a pair of the \( A \) classes. The classifier of class \( i \) against class \( j \) is defined as \( f_{ij}(x) = \text{sign}(h_{ij}(x)) \) when \( i < j \), which makes the testing example \( x' \) belong to class \( i \) if \( f_{ij}(x') = 1 \) and class \( j \) if \( f_{ij}(x') = -1 \). Besides, one can consider that \( f_{ij}(x') = -f_{ji}(x') \) when \( i > j \). The Max Wins algorithm is the most commonly used final classification rule. Make \( f_{ij}(x') = 0 \) if \( f_{ij}(x') = -1 \), the voting number of class \( i \) on \( x' \), i.e., \( D_i(x') \), is computed as \( D_i(x') = \sum_{j=1}^{A} (f_{ij}(x')) \). Then, the class which earns the highest voting number will be the final output. If more than one class earn the highest voting number, \( x' \) falls into the URs. One can also handle it by using the membership function as described in a previous study [36].

OAO has a fast training speed especially on big datasets [13]. However, the numerous classifiers tend to cause overfitting problems as they contain much overlapped information. In addition, they also lead to a dramatic increase in the evaluation time.

#### 2.2.3. Decision Directed Acyclic Graph (DDAG)

DDAG [27] adopts the same training process as OAO, whereas in the testing process, a specific graph composed of the \( A(A-1)/2 \) classifiers is constructed. Fig. 1 shows one case of the four-class problem.

The detail of the graph is determined by a pre-defined sequence of nodes, and each node represents one of the \( A(A-1)/2 \) classifiers. One can choose any pair of the classes as the root node, and the \( A-1 \) leaf nodes finally determine the output.
classes. If $f_j(x') = 1$, one can regard the testing example $x'$ as not belonging to class $j$ rather than belonging to class $i$. The classification procedure of $x'$ begins from the root node, and finally reaches to a leaf node with an output class.

As a modified structure of DDAG, ADAG [17] has a lower number of decision levels and it reduces the dependency on the sequence of nodes. These DAG architectures can handle the unclassifiable patterns of OAO. However, in an objective manner, the URs continue existing. In other words, they have not been eliminated, but rather they have been evaded during the testing process. In addition, the seeking for the best sequence of nodes is also a complicated process.

### 2.2.4. Decision Tree (DT)

The DT-based method builds up a binary tree, which makes binary decisions among the classes by the tree nodes. In order to determine the tree structure, different techniques and criteria have been adopted such as Euclidean distance [34], Mahalanobis distance [34], self-organizing-map [7], and information entropy [31]. These techniques could possibly improve the performance of the tree, but they always increase the model complexity. As the simplest solution, Fig. 2 demonstrates one case for a four-class problem. It is clear that for an $A$-class problem, the tree always contains $A - 1$ nodes.

In DT-based method, each hyperplane just exerts its partition in one subdivision of the feature space, thus there will be no UR. But when the number of classes is large, the method of designing a proper tree structure is quite intricate.

We have briefly discussed the approaches in the above section, however, there are many other methodologies and reviews on M-SVMs that may be taken into consideration [13,32,38,39,42].

### 3. Linear separability of dataset in feature space and feature space partition

This section presents some background knowledge as the basis of the VVD model, i.e., kernel trick, linear separability of dataset in feature space, and feature space partition.

#### 3.1. Kernels for SVMs

With regards to binary classification problems, SVMs aim at generating a linear hyperplane that can separate the two classes. If the two classes of data, i.e., $\{(x_i, y_i)\}_{i=1}^n \in \mathbb{R}^d \times \{+1, -1\}$, are nonlinearly separable in the input space $\mathbb{R}^d$, they are intended to be mapped into a higher dimensional feature space $\mathcal{Z}$ via a mapping function $\phi: \mathbb{R}^d \rightarrow \mathcal{Z}$, which makes them linearly separable. Kernel trick [29] represents a way to implement this procedure without having to compute the mapping $\phi$ explicitly. The mapped space $\mathcal{Z}$ is called inner product space in which the inner products of the mapped points could be replaced by functions defined on pairs of input patterns. This substitution is denoted by $\langle \phi(x_i), \phi(x_j) \rangle = k(x_i, x_j)$, where $x_i$ and $x_j$ are two input patterns, and $k(x_i, x_j)$ is a kernel function.

Two kinds of kernels are always applied to SVMs. The first kind is known as translation invariant kernel that is denoted by $k(x_i, x_j) = f(d(x_i, x_j))$ where $d(x_i, x_j) = \|x_i - x_j\| = \sqrt{\langle x_i - x_j, x_i - x_j \rangle}$. This kind of kernel is just determined by the difference between two input examples and independent of their absolute positions. The most widely used translation invariant kernel is the gaussian radial basis function (RBF) kernel, expressed as $k(x_i, x_j) = \exp(-\|x_i - x_j\|^2/2\sigma^2)$. The second kind is known as dot product kernel, which is denoted by $k(x_i, x_j) = \langle x_i, x_j \rangle$. This kind of kernel is expressed as the form of inner product, and the most widely used one is the polynomial kernel, expressed as $k(x_i, x_j) = (x_i, x_j)^d$ where $d$ represents the degree value.

Applying kernel trick to the training of SVMs always involves calculating kernel matrix defined as Definition 1.

**Definition 1** [29]. Given a function $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{K}$ (where $\mathcal{X}$ is a nonempty set, $\mathbb{K}$ is the mapped set) and patterns $x_1, \ldots, x_n \in \mathcal{X}$, the $n \times n$ matrix $K$ with elements $K_{ij} := k(x_i, x_j)$, $i, j = 1, \ldots, n$ is called the gram matrix (or kernel matrix) of $k$ with respect to $x_1, \ldots, x_n$.

![Fig. 2. DT-based structure (a case of the four-class problem).](image-url)
3.2. Linear separability of finite dataset in feature space

We now discuss the linear separability of a finite dataset in feature space. The discussion will mainly focus on gaussian RBF kernel based on which our work is proposed.

**Theorem 1** ([24,29]). Suppose that \( x_1, \ldots, x_n \in \chi \) are distinct points, and \( \sigma \neq 0 \). The matrix \( K \) given by gaussian RBF kernel
\[
K_{i,j} := \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right), \ i, j = 1, \ldots, n \text{ has full rank.}
\]

Theorem 1 is important for analyzing gaussian RBF kernel. It is easy to check that for each input pattern with this kernel, i.e., \( x_i, i = 1, \ldots, n \), we have \( k(x_i, x_i) = 1 \), which implies \( ||\phi(x_i)|| = 1 \). It means that in the feature space, all the mapped points, i.e., \( \phi(x_1), \ldots, \phi(x_n) \), are restricted to a certain area that should be a unit hypersphere. Thus, Theorem 1 shows an important property of gaussian RBF kernel: with no restriction on the number of input patterns and provided that all the input patterns are distinct, the associated mapping \( \phi \) maps them into an infinite dimensional feature space in which all mapped points are linearly independent. Obviously, Theorem 1 is a sufficient condition for \( \{x_1, \ldots, x_n\} \) being linearly separable in feature space, and this statement is also the rationale of kernel trick. Generally speaking, linearly independent implies that any binary partition of the dataset is linearly separable in feature space. For example, suppose that we abandoned the original class information of the training examples, no matter how we assign new labels (+1 or -1) to them, the training for the hyperplane should still be feasible in feature space.

Except gaussian RBF kernel, the linearly independent property of other kernels could also be analyzed as in a previous study [6].

3.3. Feature space partition

Suppose \( a \) is a finite positive integer, let us consider that with \( a \) hyperplanes in the same feature space, how many space subdivisions can we get at most? Apparently, when the dimensionality of feature space is high enough, the number of subdivisions will be at most doubled when the number of hyperplanes is increased by one.

As a general rule, for a multi-classification problem, the number of classifiers is decided by both the number of classes and its related method. When the method is fixed, a larger number of classes will lead to a larger number of classifiers. Since the number of classes for a given dataset is limited, with certain kernels, we can always assume that the number of classifiers is smaller than the dimensionality of feature space. Obviously, this assumption is correct for gaussian RBF kernel since the feature space induced by it has an infinite dimensionality.

Finally, it could be concluded that when the dimensionality of feature space is high enough, \( a \) hyperplanes will at most separate the feature space into \( 2^a \) subdivisions.

As a result, there are two observations in the existing multi-classification methods for SVMs. First, the generated hyperplanes cannot make the best use of feature space partition, since the number of space subdivisions may be much smaller than the largest possible number. Second, the meanings of some subdivisions are ambiguous, and examples within them will be considered as unclassifiable, i.e., the UR problem. These two problems motivate us to build a new model that can take full advantage of feature space partition and directly eliminate the URs.

4. Multiclass SVMs with vector-valued decision

Based on the above-mentioned discussions, we now illustrate the VVD model for \( 2^a \)-class problem and any-class problem stepwise.

4.1. Basic idea

We shall first consider a four-class problem as the simplest case. Suppose that in a fixed feature space, two hyperplanes intersect with each other and separate the whole space into four subdivisions. If the four classes are respectively locating at the four subdivisions, the given problem is solved via these two hyperplanes. Note that in Section 3, we mentioned the linear separability of dataset in feature space, which shows an important property as any binary partition of the dataset will be linearly separable with certain kernels. In this case, the above mentioned idea could be easily realized by constructing two SVM classifiers, i.e., \( f_1 \) and \( f_2 \), which are trained on the same data but with different labels as:

Class 1: \(-1\) for \( f_1 \) and \(-1\) for \( f_2 \).
Class 2: $-1$ for $f_1$ and $+$ for $f_2$.
Class 3: $+1$ for $f_1$ and $-1$ for $f_2$.
Class 4: $+1$ for $f_1$ and $+1$ for $f_2$.

This idea is illustrated on a two dimensional-space as shown in Fig. 3. Thus, given a testing example $\mathbf{x}'$, its decision is determined as a two-dimensional boolean-valued vector $f(\mathbf{x}') = (f_1(\mathbf{x}'), f_2(\mathbf{x}'))$, where $f_1(\mathbf{x}')$ and $f_2(\mathbf{x}')$ are the two trained SVM classifiers. If $f(\mathbf{x}') = (-1, -1)$: then $\mathbf{x}'$ belongs to class 1.
(-1, +1): then $\mathbf{x}'$ belongs to class 2.
(+1, -1): then $\mathbf{x}'$ belongs to class 3.
(+1, +1): then $\mathbf{x}'$ belongs to class 4.

It is clear that a two-dimensional boolean-valued vector has at most four representations, which are $(-1, -1), (-1, +1), (+1, -1), (+1, +1)$. In fact, it is not necessary to assign labels to the four classes as stated above. As long as they cover all the four representations, any assignment is feasible. In other words, we can also assign $(+1, +1)$ to class 1 and $(-1, -1)$ to class 4, instead of $(-1, -1)$ to class 1 and $(+1, +1)$ to class 4. This explanation is established on the rationale of dataset's linear separability in feature space.

This idea could be easily extended to $2^a$-class problems. Generally, we can construct $a$ SVM classifiers, i.e., $f_1, \ldots, f_a$, which separate the feature space into $2^a$ subdivisions, and each class is supposed to be lying in one subdivision. Thus, the decision of the testing example $\mathbf{x}'$ is an $a$-dimensional boolean-valued vector that is expressed as $f(\mathbf{x}') = (f_1(\mathbf{x}'), \ldots, f_a(\mathbf{x}'))$. Similarly, as long as the $2^a$ classes cover all the $2^a$ representations of an $a$-dimensional boolean-valued vector, any label assignment is feasible.

Thus, two conclusions can be drawn: (1) the $2^a$-class problem is solved by the least number of classifiers, since each class should be located at one subdivision of the feature space, and $a$ is the smallest number of hyperplanes that can separate the feature space into $2^a$ subdivisions and (2) UR does not exist, since $2^a$ is the biggest number of subdivisions that $a$ hyperplanes can separate into, and each subdivision has an explicit label.

4.2. Label assignment

In order to implement the idea, the label information of each class should be pre-processed. It is clear that for a $2^a$-class problem, the labels range from 1 to $2^a$. Each label should be transferred into an $a$-dimensional boolean-valued vector with the $m$th boolean value being the label for training the $m$th classifier where $m = 1, \ldots, a$. In this case, assigning labels to the classes is similar with designing the error-correcting output codes (ECOC) [8,9], but with distinct goals. In other words, both of them assign different codewords to different classes. The ECOC aims to produce codes with good error-correcting ability, hence, the codewords always have a high dimensionality that will lead to a large number of classifiers. However, the code-words designed for the proposed idea are with the lowest dimensionality, which will lead to the least number of classifiers.

It is inferred that different labeling orders of classes may generate different results. Here, we first discuss the assigning method when the labeling order is fixed, and we will address the labeling order issue in the next section.

Adopting the binary code is the simplest solution. Suppose that for a $2^a$-class problem, $y_i$ is the class label of the training example $\mathbf{x}_i$, and $y_i \in [1, \ldots, 2^a]$. The decimal integer $y_i - 1$ is transferred into an $a$-bit binary code, and the $m$th bit is taken as

![Fig. 3. Separate four classes with two hyperplanes.](image-url)
the class label of \(x_i\) for training the \(m\)th classifier where \(m = 1, \ldots, a\). As an example, this process for a 2\(^2\)-class problem is described in Table 1.

In brief, \(y_i\) is transferred from a decimal integer into an \(a\)-dimensional boolean-valued vector \(y_i = (y_{i1}, \ldots, y_{ia})\) for constructing the \(a\) classifiers.

### 4.3. Formulation of VVD on 2\(^a\)-class problem

Given a training set \(\mathcal{X} = \{(x_i, y_i)\}_{i=1}^n\) which contains 2\(^a\) classes, where \(a\) could be any finite positive integer, \(x_i = (x_{i1}, \ldots, x_{ia}) \in \mathbb{R}\) is the \(i\)th training example, \(y_i = (y_{i1}, \ldots, y_{ia})\) is the label vector of \(x_i\), \(y_{im} \in \{+1, -1\}\) represents the class label of \(x_i\) for the \(m\)th classifier and \(m = 1, \ldots, a\).

For training the \(m\)th classifier, we consider the training set as \(\{(x_i, y_{im})\}_{i=1}^n\), \(m = 1, \ldots, a\). If it is linearly separable, \(w_m^T x_i + b_m = 0\) exists which satisfies which satisfies that \(y_{im}(w_m^T x_i + b_m) \geq 0\), \(i = 1, \ldots, n\), \(m = 1, \ldots, a\). By adopting the formulation of (2), the problem could be concluded as (4),

\[
\begin{align*}
\min_{w_m,b_m,\xi^m} & \quad \frac{1}{2} w_m^T w_m + C_m \sum_{i=1}^n \xi_i^m \\
\text{s.t.} & \quad y_{im}(w_m^T x_i + b_m) \geq 1 - \xi_i^m, \\
& \quad \xi_i^m \geq 0, \quad i = 1, \ldots, n, \quad m = 1, \ldots, a
\end{align*}
\]

where \(C_m\) is a constant that controls the trade-off between the maximum margin and the minimum training error of the \(m\)th classifier, and \(\xi_i^m\) is a slack variable introduced to each training example for a soft margin. The Lagrange function is then constructed as (5) for each classifier,

\[
L(w_m, b_m, \xi^m, \alpha^m, \beta^m) = \frac{1}{2} w_m^T w_m + C_m \sum_{i=1}^n \xi_i^m - \sum_{i=1}^n \xi_i^m y_{im}(w_m^T x_i + b_m) - 1 + \xi_i^m - \sum_{i=1}^n \beta_i \xi_i^m
\]

where \(\xi_i^m = (\xi_{i1}^m, \xi_{i2}^m, \ldots, \xi_{ia}^m)\) are the slack variables, \(x^m = (x_{i1}^m, x_{i2}^m, \ldots, x_{ia}^m) \in \mathbb{R}^n\), \(\beta^m = (\beta_{i1}^m, \beta_{i2}^m, \ldots, \beta_{ia}^m) \in \mathbb{R}^n\), \(m = 1, 2, \ldots, a\) are Lagrange multipliers.

Based on the Karush–Kuhn–Tucker (KKT) theorem, the dual form of (5) could be formulated as (6),

\[
\begin{align*}
\max_{\alpha^m} W(\alpha^m) &= \sum_{i=1}^n \alpha_i^m - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_{im} y_{jm} \alpha_i^m \alpha_j^m k(x_i, x_j) \\
\text{s.t.} & \quad \sum_{i=1}^n y_{im} \alpha_i^m = 0 \\
& \quad \alpha_i^m \geq 0, \quad i = 1, \ldots, n, \quad m = 1, \ldots, a
\end{align*}
\]

and the \(a\) SVM classifiers are finally derived as (7),

\[
f_m(x) = \text{sign}(h_m(x)) = \text{sign} \left( \sum_{i=1}^n \alpha_i^m y_{im} k(x, x_i) + b_m \right)
\]

\(m = 1, \ldots, a\)

For the testing example \(x^*\), its decision could be expressed as (8),

\[
f(x^*) = (f_1(x^*), f_2(x^*), \ldots, f_a(x^*))
\]

### Table 1

Label assignment for an eight-class problem.

<table>
<thead>
<tr>
<th>(y_i)</th>
<th>(y_i - 1)</th>
<th>(y_i - 1)</th>
<th>(y_i - 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
and the final output class could be computed by an inverse process of the label assignment described as follows.

1. output = 0
2. for m = 1 to a
3. if \( f_m(x') = +1 \)
4. output = output + 2\(^{a-m}\)
5. end
6. end
7. output = output + 1

Obviously, when \( a = 1 \), this model degenerates back to traditional SVMs. Besides, the model could also be formulated by adopting the “all-together” method, in which all hyperplanes are derived by solving a single optimization problem as in (9).

\[
\min_{w_m, b_m, \xi_m} \frac{1}{2} \sum_{m=1}^{a} w_m^T w_m + \sum_{m=1}^{a} \sum_{i=1}^{n} C_m \xi_i^m \\
\text{s.t.} \quad y_m(x_m^T w_m + b_m) \geq 1 - \xi_i^m \\
\xi_i^m \geq 0, \quad i = 1, \ldots, n, \quad m = 1, \ldots, a
\]  

(9)

The corresponding Lagrange function is constructed as (10).

\[
L(w_1, \ldots, w_a, b_1, \ldots, b_a, \xi_1, \ldots, \xi_n, \alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n) = \sum_{m=1}^{a} \frac{1}{2} w_m^T w_m + \sum_{m=1}^{a} \sum_{i=1}^{n} C_m \xi_i^m \\
- \sum_{m=1}^{a} \sum_{i=1}^{n} \alpha_i^m [y_m(x_i^T w_m + b_m) - 1 + \xi_i^m] - \sum_{m=1}^{a} \sum_{i=1}^{n} \beta_i^m \xi_i^m
\]  

(10)

According to KKT conditions, the dual form of (10) could also be constructed, and the \( a \) hyperplanes could be achieved.

4.4. Extension of VVD to any-class problem

Let us analyze a seven-class case. Based on the idea of VVD, since \( 2^2 < 7 < 2^3 \), two classifiers cannot fully solve the problem, whereas three classifiers will generate one UR. In this case, we have defined a hierarchical structure with different classification levels. In the first level, two classifiers are constructed to separate the space into four subdivisions, and each subdivision contains at least one class. Then, classifiers of the second level are further constructed in the subdivisions with classes more than one. It is worth noting that the design of the current level will affect the latter levels. Fig. 4a and b shows two different solutions to the seven-class problem. In Fig. 4a, it is solved with five classifiers in three levels, while in Fig. 4b, it is solved with four classifiers in two levels. Since both of them do not have UR, the solution in Fig. 4b is preferred due to a simpler structure and less classifiers.

Based on this notion, we now discuss how to extend the VVD model to any-class problem. Given an \( A \)-class case where \( A \geq 3 \). Let \( a_0 \) be the total number of involved classifiers, \( p \) be the total classification levels, and \( a_i (i = 1, \ldots, p) \) be the number of classifiers in the \( i \)-th level. In order to get the simplest structure, one needs to minimize the values of \( a_0 \) and \( p \). It is worth noting that the solutions to this problem may not be unique, while one of them could be realized by making \( A \), \( p \), \( a_0 \), and \( \{a_1, \ldots, a_p\} \) satisfy the following conditions:

1. \( a_0 = \sum_{i=1}^{p} a_i \)
2. \( a_1 \geq a_2 \geq \ldots \geq a_p \)
3. \( 2^{p+1} \geq A > 2^{a_1} \)
4. \( 2^{a_j + 1} > A - \sum_{i=1}^{j-1} (2^{a_i} - 1) > 2^a, \quad j = 2, \ldots, p - 1 \)
5. \( A - \sum_{i=1}^{p-1} (2^{a_i} - 1) = 2^{a_p} \)

If \( A = 2^{a_{p+1}} \), the problem can be solved by the established model in Section 4.3. Otherwise, it is further extended to the structure shown in Fig. 5. In this structure, \( p \) levels of classifiers are designed. Each level represents a \( 2^a \)-class problem. In the \( i \)-th level, \( a_i \) classifiers are constructed which separate the remained feature space into \( 2^a \) subdivisions. The first \( 2^a - 1 \) subdivisions fix the labels of the first \( 2^a - 1 \) classes, and the last subdivision is the remained feature space that contains all the classes expected to be determined later. Besides, \( a_i \) is the largest possible number of classifiers that will not generate any UR.

One may notice that the structure in Fig. 5 is similar to that of DT as both of them can solve the problem via a sequence of decisions. However, in DT-based method, each node represents one classifier that makes binary decisions among the classes, whereas in the proposed VVD model, each level is composed of several classifiers that makes multiple decisions.

As a result, VVD solves the seven-class problem with only four classifiers. Obviously, this is the simplest model one can achieve, because larger numbers of classifiers tend to generate more complex models, while the numbers of classifiers constructed by OAA, OAO, DDAG, and DT are respectively 7, 21, 21, and 6.
5. Related discussions

5.1. Effects of unbalanced data on the VVD model

It is concerned that unbalanced datasets may affect the performance of the VVD model. Unbalanced refers to the case that examples in training set belonging to one class heavily outnumber the examples in the other class. We select some methods to balance the class distribution and conduct experiments to investigate the performance of the model on balanced and unbalanced data respectively.

Generally, over-sampling of minority class and under-sampling of majority class are the two main approaches for balancing data. Ten related methods have been reviewed in a previous study [2], we select three of them in the experiments.
Random over-sampling: This method balances the class distribution by randomly replicating the minority class examples.

Random under-sampling: This method balances the class distribution by randomly eliminating the majority class examples.

Synthetic Minority Over-sampling Technique (SMOTE): The main idea of this method is to over-sample the minority class by creating synthetic examples instead of replicating. It creates extra training data by interpolating between several minority class examples that are close to each other. Detailed descriptions and analysis could be found in another study [5].

Normally, the above-mentioned methods are used for processing binary datasets. Thus, in the multiclass case, the classes with the largest number of examples are treated as the majority classes, and the others are taken as the minority ones. An illustration of the over- and under-sampling results are shown in Fig. 6. Four classes are artificially generated in a two-dimensional space, which could be separated by two SVM hyperplanes. Since the examples in the four classes are unbalanced, the three methods could be performed. It is observed that with the over- and under-sampling methods, the trained classifiers are slightly different from the original ones.

In order to look into this impact, we conduct some further investigations on ten University of California at Irvine (UCI) machine learning datasets. Four classes of examples from each of the ten datasets were used. By randomly selecting examples from each class, the datasets become unbalanced as listed in Table 2. Note that datasets zoo, glass, derm and chart were conducted in a ten-fold cross-validation manner. The VVD model was performed on the original unbalanced data, the data with random over-sampling, random under-sampling, and SMOTE. The SVM parameter C was fixed as 100, gaussian RBF kernel was adopted with the kernel parameter as \( \gamma = \{2^{-17}, 2^{-16}, \ldots, 2^2\} \).

Except testing accuracy, the area under the ROC curve (AUC) is also considered as a metric for measuring performance. Note that ROC is originally designed for measuring binary classification results by calculating the true positive and false positive rates of the confusion matrix. In our comparisons, we adopt multiclass AUC by measuring the unweighted pairwise discriminability of classes [10] as in the following equation:

$$AUC_{total} = \frac{2}{|X'|(|X'| - 1)} \sum_{X'_i, X'_j \in X'} AUC(X'_i, X'_j)$$ (11)

where $X'$ represents the whole testing set, $|X'|$ is the number of classes and $AUC(X'_i, X'_j)$ is the two-class testing AUC involving class $i$ and class $j$.

The probabilistic outputs of the testing examples should be first computed for calculating the AUC values. For traditional SVMs, the probabilistic outputs of the testing example $x$ are always computed by the logistic regression model, i.e., $p(y = 1|x) = 1/(1 + e^{-h(x)})$ and $p(y = -1|x) = 1 - p(y = 1|x)$. This calculation is more complicated with regard to the VVD model. Due to the pairwise mechanism, we need to know the values of $p(y = i|x)$ and $p(y = j|x)$ when calculating $AUC(X'_i, X'_j)$, where $p(y = i|x) * p(y = j|x) = 1$. Given that $p_m(y = 1|x) = 1/(1 + e^{-h_m(x)})$ and $p_m(y = -1|x) = 1 - p_m(y = 1|x)$ are the probabilistic outputs of $x$ regarding the $m$th classifier, $(y'_1, \ldots, y'_n)$ and $(y'_1, \ldots, y'_n)$ are the assigned label vectors of class $i$ and class $j$, then, $p(y = i|x)$ and $p(y = j|x)$ could be calculated as (12) and (13).

$$p(y = i|x) = \frac{\prod_{m=1}^n P_m(y = y'_m|x)}{\prod_{m=1}^n P_m(y = y'_m|x) + \prod_{m=1}^n P_m(y = y'_m|x)}$$ (12)

$$p(y = j|x) = \frac{\prod_{m=1}^n P_m(y = y'_m|x)}{\prod_{m=1}^n P_m(y = y'_m|x) + \prod_{m=1}^n P_m(y = y'_m|x)}$$ (13)

The maximum and mean values of testing accuracy and AUC on the ten datasets are reported in Fig. 7. It can be seen that for both testing accuracy and AUC, the maximum values on the unbalanced data and balanced data have no significant difference. This phenomenon also appears in the mean AUC values, which shows that the unbalanced datasets may affect the performance of the VVD model, however, its impact is not significant.

### 5.2. Labeling order of classes

Labeling order of classes is another important issue that may affect the performance of the model. To investigate this impact, we need to determine the total number of labeling orders for a given $2^n$-class problem. The key points for determining this number are listed as follows:
Fig. 7. Statistics of testing accuracy and AUC of unbalanced data.

(a) Maximum testing accuracy of each unbalanced dataset

(b) Mean testing accuracy of each unbalanced dataset

(c) Maximum AUC value of each unbalanced dataset

(d) Mean AUC value of each unbalanced dataset

Fig. 8. Three different labeling orders of a four-class problem.
(1) For each of the classifiers, half of the classes are taken as positive, and the other half as negative.

(2) The first classifier can take any $2^{n-1}$ classes as positive, while the second classifier needs to select $2^{n-2}$ classes from each of the two parts of the first classifier’s result, and the third classifier needs to select $2^{n-3}$ classes from each of the four parts of the second classifier’s result, etc.

(3) For each classifier, the positive and negative labels could be exchanged, and this action makes no difference to the training.

(4) The classifiers could be with any order, and the result should not be affected by their permutation.
By taking these four points into consideration, the number of labeling orders for \(2^a\)-class problem is then decided using the following equation:

\[
num = \left(\prod_{m=0}^{a-1} \left[\left(\frac{2^a-1}{2}\right)^2 / 2\right] \right) / \left(\prod_{m=0}^{a-1} \left[\left(\frac{2^a-m}{(2^a-1)}\right)^2 / 2\right] \right) = \frac{a!}{a^a} = \frac{a!}{a^a} = \frac{a!}{a^a} = \frac{a!}{a^a}.
\]

(14)

where \(C\) represents the combination, and \(P\) represents the permutation.

It can be seen from (14) that when \(a\) is updated from 2 to 3, the \(num\) is changed from 3 to 840. If we want to compare all the labeling orders, it has already become impossible for \(a = 3\). In this case, we only make some efforts to discuss the four-class problem. The spatial relationships among the four classes and the two hyperplanes of the three different labeling orders have been roughly illustrated in Fig. 8.

We used the original four classes of examples listed in Table 2 for this comparison, since both balanced and unbalanced data are contained. The parameter settings are exactly the same as in Section 5.1, and the results are shown in Fig. 9. It is observed that in most cases, different labeling orders generate different results. However, this difference is so small that the overall trends are still maintained.

Since there exits numerous labeling orders, we apply the similarity-based techniques for simplicity reason. Our basic assumptions are that two classes of examples with smaller similarity are easy to separate, thus they can be handled by less classifiers. In contrast, two classes of examples with larger similarity need to be handled by more classifiers. In the \(2^a\)-class problem, the \(i\)th class and the \((2^a-i+1)\)th class are always in the corresponding positions which will be handled by \(a\) classifiers, while neighboring classes will be handled by one classifier only (see Table 1 as an example). Our ordering method is described in Algorithm 1.

**Algorithm 1:** Ordering classes for a \(2^a\)-class problem

1: Randomly select a class, and label it as 1;
2: For \(i = 1\) to \((2^a - 1 - 1)\)
3: Calculate the similarities between class \(i\) and all the rest classes;
4: Select the class with the smallest similarity and label it as \(i + 1\);
5: Select the class with the largest similarity and label it as \(2^a - i + 1\);
6: EndFor
7: Label the rest class as \(2^a - i + 1\).

In addition, the similarity between two classes could be measured by gaussian kernel, as it is well known that gaussian kernel also serves as a similarity relation. Given that \(X_i\) and \(X_j\) are the two referred classes of training data, their similarity is then computed as \(\sum_{x_i \in X_i, x_j \in X_j} \exp(-||x_i - x_j||^2/2\sigma^2)/||x_i|| \cdot ||x_j||\), where \(|x_i|\) and \(|x_j|\) represent the numbers of training examples in class \(i\) and class \(j\). Besides, for non-\(2^a\)-class problems, Algorithm 1 is performed in each level.

### 6. Experimental comparisons

#### 6.1. Experimental settings

In this section, experiments are conducted to validate the effectiveness of the proposed model. The OAA, OAO, DDAG, and DT methods introduced in Section 2, together with the proposed VVD model, were compared on ten UCI datasets. Besides, both the original class order and an order designed by Algorithm 1 were considered. The details of the selected datasets are listed in Table 3. Datasets zoo, glass, derm, chart, and soybean are small ones that contain examples less than 5000; datasets opt, satellite, pen, letter, and shuttle are larger ones that contain examples more than 5000.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Train example</th>
<th># Test example</th>
<th># Attribute</th>
<th># Class (org)</th>
<th># Class ((2^a))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zoo</td>
<td>101</td>
<td>10-fold</td>
<td>17</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>Glass</td>
<td>214</td>
<td>10-fold</td>
<td>11</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Derm</td>
<td>366</td>
<td>10-fold</td>
<td>35</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Chart</td>
<td>600</td>
<td>10-fold</td>
<td>61</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Soybean</td>
<td>307</td>
<td>376</td>
<td>36</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>Opt</td>
<td>3823</td>
<td>1797</td>
<td>65</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Satellite</td>
<td>4435</td>
<td>2000</td>
<td>37</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>Pen</td>
<td>7494</td>
<td>3498</td>
<td>17</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Letter</td>
<td>15,000</td>
<td>5000</td>
<td>17</td>
<td>26</td>
<td>16</td>
</tr>
<tr>
<td>Shuttle</td>
<td>43,500</td>
<td>14,500</td>
<td>10</td>
<td>7</td>
<td>4</td>
</tr>
</tbody>
</table>
Table 4

2-class problem: optimal parameter, mean accuracy and standard deviation of the top ten classification models based on validation result.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>OAA ( (\gamma, C) )</th>
<th>Rate (%)</th>
<th>OAO ( (\gamma, C) )</th>
<th>Rate (%)</th>
<th>DDAG ( (\gamma, C) )</th>
<th>Rate (%)</th>
<th>DT ( (\gamma, C) )</th>
<th>Rate (%)</th>
<th>VVD ( (\gamma, C) )</th>
<th>Rate (%)</th>
<th>Ordered VVD ( (\gamma, C) )</th>
<th>Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zoo</td>
<td>( (2^{4}, 2^{5}) )</td>
<td>98.79 ± 0.72</td>
<td>( (2^{4}, 2^{5}) )</td>
<td>98.71 ± 0.76</td>
<td>( (2^{4}, 2^{5}) )</td>
<td>98.71 ± 0.76</td>
<td>( (2^{4}, 2^{5}) )</td>
<td>98.70 ± 0.77</td>
<td>( (2^{4}, 2^{5}) )</td>
<td>99.14 ± 0.67</td>
<td>( (2^{4}, 2^{5}) )</td>
<td>99.09 ± 0.00</td>
</tr>
<tr>
<td>glass</td>
<td>( (2^{-11}, 2^{12}) )</td>
<td>100.00 ± 0.00</td>
<td>( (2^{-17}, 2^{12}) )</td>
<td>100.00 ± 0.00</td>
<td>( (2^{-17}, 2^{12}) )</td>
<td>100.00 ± 0.00</td>
<td>( (2^{-17}, 2^{12}) )</td>
<td>100.00 ± 0.00</td>
<td>( (2^{-11}, 2^{13}) )</td>
<td>100.00 ± 0.00</td>
<td>( (2^{-11}, 2^{10}) )</td>
<td>100.00 ± 0.00</td>
</tr>
<tr>
<td>derm</td>
<td>( (2^{-16}, 2^{13}) )</td>
<td>97.66 ± 0.13</td>
<td>( (2^{-14}, 2^{14}) )</td>
<td>97.94 ± 0.18</td>
<td>( (2^{-14}, 2^{14}) )</td>
<td>97.94 ± 0.18</td>
<td>( (2^{-14}, 2^{14}) )</td>
<td>97.93 ± 0.18</td>
<td>( (2^{-15}, 2^{13}) )</td>
<td>97.46 ± 0.18</td>
<td>( (2^{-16}, 2^{14}) )</td>
<td>97.04 ± 0.21</td>
</tr>
<tr>
<td>chart</td>
<td>( (2^{-11}, 2^{12}) )</td>
<td>99.82 ± 0.05</td>
<td>( (2^{-12}, 2^{6}) )</td>
<td>99.83 ± 0.00</td>
<td>( (2^{-12}, 2^{6}) )</td>
<td>99.83 ± 0.00</td>
<td>( (2^{-12}, 2^{6}) )</td>
<td>99.83 ± 0.00</td>
<td>( (2^{-11}, 2^{7}) )</td>
<td>99.78 ± 0.08</td>
<td>( (2^{-11}, 2^{7}) )</td>
<td>99.93 ± 0.09</td>
</tr>
<tr>
<td>soybean</td>
<td>( (2^{-16}, 2^{14}) )</td>
<td>93.83 ± 0.37</td>
<td>( (2^{-4}, 2^{3}) )</td>
<td>94.79 ± 0.22</td>
<td>( (2^{-4}, 2^{3}) )</td>
<td>94.26 ± 0.44</td>
<td>( (2^{-4}, 2^{3}) )</td>
<td>94.30 ± 1.01</td>
<td>( (2^{-5}, 2^{3}) )</td>
<td>92.82 ± 1.03</td>
<td>( (2^{-5}, 2^{3}) )</td>
<td>92.34 ± 0.72</td>
</tr>
<tr>
<td>opt</td>
<td>( (2^{-10}, 2^{7}) )</td>
<td>98.55 ± 0.00</td>
<td>( (2^{-11}, 2^{6}) )</td>
<td>98.12 ± 0.02</td>
<td>( (2^{-11}, 2^{6}) )</td>
<td>98.45 ± 0.02</td>
<td>( (2^{-10}, 2^{7}) )</td>
<td>98.39 ± 0.00</td>
<td>( (2^{-9}, 2^{7}) )</td>
<td>98.33 ± 0.00</td>
<td>( (2^{-9}, 2^{7}) )</td>
<td>98.50 ± 0.00</td>
</tr>
<tr>
<td>satellite</td>
<td>( (2^{-10}, 2^{7}) )</td>
<td>91.55 ± 0.21</td>
<td>( (2^{-11}, 2^{6}) )</td>
<td>91.54 ± 0.25</td>
<td>( (2^{-11}, 2^{6}) )</td>
<td>91.49 ± 0.23</td>
<td>( (2^{-11}, 2^{6}) )</td>
<td>91.28 ± 0.26</td>
<td>( (2^{-11}, 2^{6}) )</td>
<td>91.29 ± 0.05</td>
<td>( (2^{-11}, 2^{6}) )</td>
<td>91.40 ± 0.00</td>
</tr>
<tr>
<td>pen</td>
<td>( (2^{-11}, 2^{6}) )</td>
<td>98.53 ± 0.09</td>
<td>( (2^{-11}, 2^{6}) )</td>
<td>98.30 ± 0.09</td>
<td>( (2^{-11}, 2^{6}) )</td>
<td>98.33 ± 0.11</td>
<td>( (2^{-11}, 2^{6}) )</td>
<td>98.48 ± 0.13</td>
<td>( (2^{-12}, 2^{6}) )</td>
<td>97.86 ± 0.08</td>
<td>( (2^{-12}, 2^{6}) )</td>
<td>98.64 ± 0.05</td>
</tr>
<tr>
<td>letter</td>
<td>( (2^{-11}, 2^{6}) )</td>
<td>97.78 ± 0.11</td>
<td>( (2^{-3}, 2^{2}) )</td>
<td>97.80 ± 0.02</td>
<td>( (2^{-3}, 2^{2}) )</td>
<td>97.69 ± 0.01</td>
<td>( (2^{-3}, 2^{2}) )</td>
<td>96.71 ± 0.07</td>
<td>( (2^{-3}, 2^{2}) )</td>
<td>95.95 ± 0.02</td>
<td>( (2^{-3}, 2^{12}) )</td>
<td>95.97 ± 0.07</td>
</tr>
<tr>
<td>shuttle</td>
<td>( (2^{-13}, 2^{6}) )</td>
<td>99.87 ± 0.02</td>
<td>( (2^{-13}, 2^{6}) )</td>
<td>99.87 ± 0.02</td>
<td>( (2^{-13}, 2^{6}) )</td>
<td>99.88 ± 0.02</td>
<td>( (2^{-13}, 2^{6}) )</td>
<td>99.87 ± 0.01</td>
<td>( (2^{-12}, 2^{10}) )</td>
<td>99.87 ± 0.01</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>avg.</td>
<td>97.64 ± 0.17</td>
<td>\textbf{97.69 ± 0.16}</td>
<td>97.66 ± 0.18</td>
<td>\textbf{97.69 ± 0.16}</td>
<td>97.45 ± 0.24</td>
<td>97.25 ± 0.21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The bold values represent the highest testing accuracy achieved by the several methods. The up and down arrows in the last column demonstrate that compared with the original VVD method, the accuracy achieved by ordered VVD is higher or lower.
Table 5
2nd-Class problem: number of classifiers, total number of SVs, and CPU time evaluation.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>OAA</th>
<th>OAO</th>
<th>DDAG</th>
<th>DT</th>
<th>VVD</th>
<th>Ordered VVD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># f</td>
<td># SVs</td>
<td>Training</td>
<td># f</td>
<td># SVs</td>
<td>Training</td>
</tr>
<tr>
<td>Zoo</td>
<td>4</td>
<td>148</td>
<td>0.060</td>
<td>6</td>
<td>146</td>
<td>0.063</td>
</tr>
<tr>
<td>Glass</td>
<td>4</td>
<td>364</td>
<td>0.219</td>
<td>6</td>
<td>281</td>
<td>0.131</td>
</tr>
<tr>
<td>Derm</td>
<td>4</td>
<td>736</td>
<td>0.627</td>
<td>6</td>
<td>649</td>
<td>0.330</td>
</tr>
<tr>
<td>Chart</td>
<td>4</td>
<td>1587</td>
<td>2.185</td>
<td>6</td>
<td>1239</td>
<td>0.870</td>
</tr>
<tr>
<td>Soybean</td>
<td>16</td>
<td>1655</td>
<td>0.112</td>
<td>120</td>
<td>2652</td>
<td>0.164</td>
</tr>
<tr>
<td>Opt</td>
<td>8</td>
<td>17,349</td>
<td>0.202</td>
<td>28</td>
<td>16,356</td>
<td>4.575</td>
</tr>
<tr>
<td>Satellite</td>
<td>4</td>
<td>11,423</td>
<td>0.112</td>
<td>120</td>
<td>26,522</td>
<td>12.125</td>
</tr>
<tr>
<td>Pen</td>
<td>8</td>
<td>36,951</td>
<td>0.202</td>
<td>28</td>
<td>33,096</td>
<td>5.148</td>
</tr>
<tr>
<td>Letter</td>
<td>16</td>
<td>73,985</td>
<td>0.112</td>
<td>120</td>
<td>108,458</td>
<td>11,353</td>
</tr>
<tr>
<td>Shuttle</td>
<td>4</td>
<td>49,779</td>
<td>0.202</td>
<td>6</td>
<td>36,903</td>
<td>664,829</td>
</tr>
</tbody>
</table>

Note: The bold values represent the least number of classifiers, the least number of SVs, the lowest training and testing time achieved by the several methods. Note that if the number of SVs and the execution time is very close to the optimal one achieved, they are also in bold font.
Two sets of experiments were performed respectively on $2^n$-class problems and non-$2^n$-class problems. First, by merging some classes, and keeping a balance among the numbers of examples, each dataset was transferred into $2^c$ classes with the largest possible value of $c$ for the experiment in Section 6.2, and then, the original data were used for the experiment in Section 6.3.

Gaussian RBF kernel was adopted for all methods. The parameters, i.e., $(\gamma, C)$, were set as $\gamma = [2^{-17}, 2^{-16}, \ldots, 2^2]$ and $C = [2^{-5}, 2^{-4}, \ldots, 2^{14}]$. Therefore, we tried $20 \times 20 = 400$ combinations of $(\gamma, C)$ on each dataset. For datasets soybean, opt, satellite, pen, and shuttle, both the training and testing sets are available. The examples in the training set were divided into two subsets (70% and 30%) for training and testing of the model validation. Then, the best ten pairs of $(\gamma, C)$ that achieved the highest validation rates were used to train the final models on the whole training set. Finally, the mean prediction rate and the standard deviation of the final models on the testing set were achieved. For datasets zoo, glass, derm, and chart, the testing sets are not available. Therefore, we adopted ten-fold cross validation on the whole training set and reported the mean value and the standard deviation of the ten highest validation rates.

We adopted the “svmtrain” and “svmpredict” mex functions of libsvm compiled by Microsoft’s Visual C++ 6.0, and performed the experiments under MATLAB 7.9.0. The SVM type was fixed as C-SVC. The experiments were executed on a computer with a 3.16-GHz Intel Core 2 Duo CPU, a maximum 4.00-GB memory, and 64-bit windows7 system.

It is noteworthy that, when ordering the classes of dataset shuttle, MATLAB was always out of memory with our ordinary PC when calculating the gaussian kernel matrix due to the large training set size, therefore, we only applied the VVD model to the original class order for this dataset.

6.2. Experiments on $2^n$-class problems

Table 6 reports the optimal parameters for $2^n$-class problems, together with the mean testing accuracy and standard deviation of the top ten models. It is easy to observe that the optimal parameters for different classification tasks are in quite different ranges. However, by sufficiently tuning $(\gamma, C)$, all methods can achieve similar generalization capabilities. In fact, it is found that except dataset soybean, the differences among the accuracies of the five related methods are all <2%. Under this condition, the comparison of testing accuracy is of no crucial significance, and our concentration may further focus on other criteria such as the model complexity or the execution time.

Table 5 reports the number of classifiers generated by each method, which is denoted by “# $f$”, and we roughly consider that a larger number of involved classifiers may lead to a more complex model. It is obvious that the proposed method is the best based on this criterion. In fact, for a $2^n$-class problem, OAA generates $2^n$ classifiers, OAO and DDAG generate $2^n(2^n - 1)/2$ ones, DT involves $2^n - 1$ ones, whereas VVD only generates $a$ ones.

Followed by the number of classes, we also analyze the total number of SVs, the training and testing time. With respect to these items, it has been investigated that different parameter settings will lead to different results for the same method on the same dataset. Thus, we tried the 400 parameter combinations and report the averaged value.

The total number of SVs, denoted by “# SVs”, is reported in Table 5. Since a training example may be a SV for more than one hyperplane with different non-zero values, thus the total number of SVs may be larger than the total number of training examples. We can see that VVD involves the least number of SVs on almost all datasets.

Table 5 also reports the training and testing time that are denoted by “training” and “testing”. Theoretically, the training cost for each classifier could be calculated according to a power law $T = cn^\lambda$, where $n$ is the training set size, $\lambda$ is determined by the training algorithm, and $c$ is a constant value. Thus, when all the classes have the same number of examples, it could be calculated that $T_{\text{OAA}} = c \times 2^n \times n$, $T_{\text{OAO}} = T_{\text{DDAG}} = c \times [2^2 \times (2^2 - 1)/2] \times (2n/2^2)^\lambda$, $T_{\text{DT}} = \sum_{i=1}^{2^n} (c \times (i \times n/2^2)^\lambda)$, $T_{\text{VVD}} = c \times a \times n$. But when it comes to practice, the class distribution is always unbalanced, and the above calculations cannot be satisfied. From Table 5, we see that OAA is the most time consuming method on all the datasets; OAO and DDAG are the best on datasets chart, opt, satellite, pen, and letter; DT is the best on dataset shuttle; VVD is the best on datasets zoo, glass, derm, and soybean. Generally speaking, for the big datasets, OAO-based methods can achieve the shortest training time, and for the small datasets, the proposed model is more efficient. In fact, OAA generates a comparatively large number of classifiers, and training each classifier involves all examples, thus it is the most costly method; for OAO and DDAG, the number of classifiers is even larger, but each one is trained by solving a small problem with two classes; and for VVD, training each classifier also involves all the examples, while the number of classifiers is quite small.

Regarding the testing expenses, the advantage of the proposed model becomes more obvious. It is worth noting that in some real-world cases, the testing time is a very important criterion to evaluate the efficiency, especially for some online classification systems, since the training process is always finished off-line, whereas the testing process needs to be done
Table 7
Non-2-class problem: Optimal parameter, mean accuracy and standard deviation of the top ten classification models based on validation result.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>OAA</th>
<th>Rate (%)</th>
<th>OAO</th>
<th>Rate (%)</th>
<th>DDAG</th>
<th>Rate (%)</th>
<th>DT</th>
<th>Rate (%)</th>
<th>VVD</th>
<th>Rate (%)</th>
<th>Ordered VVD</th>
<th>Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>zoo</td>
<td>(2^{-2}, 2^{11})</td>
<td>98.13 ± 0.11</td>
<td>(2^{-5}, 2^{12})</td>
<td>98.14 ± 0.07</td>
<td>(2^{-5}, 2^{12})</td>
<td>98.14 ± 0.07</td>
<td>(2^{-2}, 2^{13})</td>
<td>98.16 ± 0.09</td>
<td>(2^{-3}, 2^{6})</td>
<td>97.64 ± 0.43</td>
<td>(2^{-2}, 2^{7})</td>
<td>98.67 ± 0.44</td>
</tr>
<tr>
<td>glass</td>
<td>(2^{-9}, 2^{8})</td>
<td>99.78 ± 0.23</td>
<td>(2^{-9}, 2^{8})</td>
<td>99.57 ± 0.00</td>
<td>(2^{-9}, 2^{8})</td>
<td>99.57 ± 0.00</td>
<td>(2^{-9}, 2^{8})</td>
<td>99.74 ± 0.22</td>
<td>(2^{-9}, 2^{10})</td>
<td>100.0 ± 0.00</td>
<td>(2^{-9}, 2^{10})</td>
<td>99.68 ± 0.22</td>
</tr>
<tr>
<td>derm</td>
<td>(2^{-12}, 2^{4})</td>
<td>97.86 ± 0.11</td>
<td>(2^{-10}, 2^{5})</td>
<td>97.68 ± 0.19</td>
<td>(2^{-10}, 2^{5})</td>
<td>97.68 ± 0.19</td>
<td>(2^{-14}, 2^{10})</td>
<td>97.81 ± 0.17</td>
<td>(2^{-10}, 2^{6})</td>
<td>96.95 ± 0.21</td>
<td>(2^{-13}, 2^{6})</td>
<td>97.14 ± 0.38</td>
</tr>
<tr>
<td>chart</td>
<td>(2^{-11}, 2^{2})</td>
<td>99.83 ± 0.00</td>
<td>(2^{-12}, 2^{10})</td>
<td>99.85 ± 0.05</td>
<td>(2^{-12}, 2^{10})</td>
<td>99.85 ± 0.05</td>
<td>(2^{-12}, 2^{10})</td>
<td>99.83 ± 0.05</td>
<td>(2^{-16}, 2^{10})</td>
<td>99.83 ± 0.00</td>
<td>(2^{-16}, 2^{10})</td>
<td>99.83 ± 0.00</td>
</tr>
<tr>
<td>soybean</td>
<td>(2^{-5}, 2^{2})</td>
<td>93.59 ± 0.20</td>
<td>(2^{-5}, 2^{2})</td>
<td>94.84 ± 0.14</td>
<td>(2^{-5}, 2^{2})</td>
<td>94.84 ± 0.14</td>
<td>(2^{-5}, 2^{2})</td>
<td>93.59 ± 1.54</td>
<td>(2^{-5}, 2^{1})</td>
<td>91.57 ± 1.00</td>
<td>(2^{-5}, 2^{3})</td>
<td>92.85 ± 0.96</td>
</tr>
<tr>
<td>opt</td>
<td>(2^{-11}, 2^{3})</td>
<td>97.89 ± 0.02</td>
<td>(2^{-10}, 2^{3})</td>
<td>98.33 ± 0.00</td>
<td>(2^{-10}, 2^{3})</td>
<td>98.33 ± 0.00</td>
<td>(2^{-10}, 2^{3})</td>
<td>97.87 ± 0.39</td>
<td>(2^{-9}, 2^{1})</td>
<td>97.94 ± 0.00</td>
<td>(2^{-9}, 2^{10})</td>
<td>98.18 ± 0.11</td>
</tr>
<tr>
<td>satellite</td>
<td>(2^{-10}, 2^{1})</td>
<td>97.80 ± 0.10</td>
<td>(2^{-10}, 2^{1})</td>
<td>91.89 ± 0.10</td>
<td>(2^{-10}, 2^{1})</td>
<td>91.89 ± 0.10</td>
<td>(2^{-10}, 2^{1})</td>
<td>91.01 ± 0.28</td>
<td>(2^{-10}, 2^{2})</td>
<td>91.55 ± 0.08</td>
<td>(2^{-11}, 2^{2})</td>
<td>91.49 ± 0.13</td>
</tr>
<tr>
<td>pen</td>
<td>(2^{-12}, 2^{4})</td>
<td>98.42 ± 0.09</td>
<td>(2^{-13}, 2^{5})</td>
<td>98.35 ± 0.02</td>
<td>(2^{-15}, 2^{5})</td>
<td>97.90 ± 0.25</td>
<td>(2^{-15}, 2^{5})</td>
<td>98.26 ± 0.11</td>
<td>(2^{-11}, 2^{1})</td>
<td>97.68 ± 0.12</td>
<td>(2^{-11}, 2^{1})</td>
<td>98.21 ± 0.06</td>
</tr>
<tr>
<td>letter</td>
<td>(2^{-4}, 2^{2})</td>
<td>97.36 ± 0.06</td>
<td>(2^{-5}, 2^{4})</td>
<td>97.43 ± 0.05</td>
<td>(2^{-4}, 2^{4})</td>
<td>97.52 ± 0.00</td>
<td>(2^{-3}, 2^{4})</td>
<td>96.25 ± 0.04</td>
<td>(2^{-4}, 2^{1})</td>
<td>94.91 ± 0.06</td>
<td>(2^{-4}, 2^{1})</td>
<td>95.33 ± 0.05</td>
</tr>
<tr>
<td>shuttle</td>
<td>(2^{-17}, 2^{11})</td>
<td>99.88 ± 0.01</td>
<td>(2^{-16}, 2^{11})</td>
<td>99.88 ± 0.01</td>
<td>(2^{-15}, 2^{14})</td>
<td>99.88 ± 0.01</td>
<td>(2^{-15}, 2^{12})</td>
<td>99.87 ± 0.02</td>
<td>(2^{-15}, 2^{14})</td>
<td>99.90 ± 0.02</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>avg.</td>
<td>97.45 ± 0.10</td>
<td><strong>97.60 ± 0.06</strong></td>
<td>97.48 ± 0.08</td>
<td>97.24 ± 0.29</td>
<td>96.80 ± 0.19</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
</tbody>
</table>

Note: The bold values represent the highest testing accuracy achieved by the several methods. The up and down arrows in the last column demonstrate that compared with the original VVD method, the accuracy achieved by ordered VVD is higher or lower.
Table 8
Non-2<sup>nd</sup>-class problem: number of classifiers, total number of SVs, and CPU time evaluation.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>OAA</th>
<th>OAO</th>
<th>DDAG</th>
<th>DT</th>
<th>VVD</th>
<th>Ordered VVD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># f</td>
<td># SVs</td>
<td>Training</td>
<td># f</td>
<td># SVs</td>
<td>Training</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>testing</td>
<td></td>
<td></td>
<td>testing</td>
</tr>
<tr>
<td>zoo</td>
<td>7</td>
<td>199</td>
<td>0.088</td>
<td>21</td>
<td>257</td>
<td>0.157</td>
</tr>
<tr>
<td></td>
<td>0.015</td>
<td></td>
<td>0.078</td>
<td></td>
<td></td>
<td>0.043</td>
</tr>
<tr>
<td></td>
<td>0.029</td>
<td></td>
<td>0.119</td>
<td></td>
<td></td>
<td>0.080</td>
</tr>
<tr>
<td></td>
<td>0.852</td>
<td></td>
<td>0.272</td>
<td></td>
<td></td>
<td>0.184</td>
</tr>
<tr>
<td></td>
<td>0.511</td>
<td></td>
<td>0.595</td>
<td></td>
<td></td>
<td>0.462</td>
</tr>
<tr>
<td></td>
<td>1909</td>
<td>0.801</td>
<td>0.671</td>
<td>171</td>
<td>3201</td>
<td>0.156</td>
</tr>
<tr>
<td></td>
<td>0.145</td>
<td></td>
<td>0.155</td>
<td></td>
<td></td>
<td>0.151</td>
</tr>
<tr>
<td></td>
<td>21,347</td>
<td>23.132</td>
<td>45</td>
<td>21,222</td>
<td>4.298</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>26.734</td>
<td></td>
<td>27.559</td>
<td></td>
<td></td>
<td>10.454</td>
</tr>
<tr>
<td></td>
<td>16,496</td>
<td>14.439</td>
<td>21</td>
<td>14,003</td>
<td>8.629</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>45,969</td>
<td>201.947</td>
<td>45</td>
<td>42,686</td>
<td>20.716</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>42.441</td>
<td></td>
<td>41.180</td>
<td></td>
<td></td>
<td>52.934</td>
</tr>
<tr>
<td></td>
<td>82.870</td>
<td></td>
<td>145.604</td>
<td></td>
<td></td>
<td>166.109</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>67,436</td>
<td>1399.071</td>
<td>21</td>
<td>59,478</td>
<td>800.190</td>
</tr>
<tr>
<td></td>
<td>230.324</td>
<td></td>
<td>207.530</td>
<td></td>
<td></td>
<td>309.161</td>
</tr>
</tbody>
</table>

Note: The bold values represent the least number of classifiers, the least number of SVs, the lowest training and testing time achieved by the several methods. Note that if the number of SVs and the execution time is very close to the optimal one achieved, they are also in bold font.
on-line. Depending on the model structure, the testing expense of a method is mainly decided by two factors: (1) kernel evaluation and (2) additional operation. One observes from Table 5 that VVD costs the shortest testing time on all the datasets. This advantage is possibly benefited from a smaller number of kernel evaluations and a simpler post-processing task. On one hand, if there exists some URs, the learner needs to make some additional efforts to handle the unclassifiable examples. Obviously, OAA and OAO-based methods suffer from this problem, while one of the most important characteristics of VVD is that it contains no UR. On the other hand, the testing processes of all the other four methods involve some additional operations, such as the comparing or sorting process of OAA, the voting process of OAO, the graph or tree structure formation of DDAG and DT, while VVD only performs an inverse process of the label assignment by adding up some integers.

Besides, compared with the original class order, the implementation of Algorithm 1 leads to some performance improvements of five datasets out of nine. However, the reported results of ordered VVD may not be optimal, since the first class is still selected randomly in Algorithm 1. Thus, the performance of VVD could be further improved if the optimal order is found, and this is another important research topic we aim to do in the future.

Finally, Table 6 reports the $p$-values of statistical tests on the testing accuracies in Table 4. We adopted paired Wilcoxon signed rank test, since it is a non-parametric test that could be performed on both small and large samples, and does not require normal distribution. For dataset shuttle with ordered VVD, the accuracy of VVD was used. Usually, the confidence of the test is 95%, which means that the results are significantly different if the $p$-value is <5%. Thus, Table 6 further confirms that these methods are in fact of no essential difference in term of testing accuracy. Therefore, the way to improve the learning efficiency will be the most important issue of SVM-based multi-classification problem.

6.3. Experiments on non-$2^n$-class problems

Tables 7–9 report the results of non-$2^n$-class problems, which are roughly consistent with those of $2^n$-class problems. The accuracies achieved by the several methods are of no essential difference. While VVD gives this outcome by involving a smaller number of SVs, a medium training time, and a shorter testing time on most of the selected datasets. Besides, the class order guided by Algorithm 1 leads to some performance improvements of six datasets out of nine.

7. Conclusion

In SVM-based multi-classification tasks, most of the existing methods suffer from the intensive computation and UR problems. This paper proposes a new model named VVD, which solves a $2^n$-class problem by a SVM hyperplanes in the feature space. Wilcoxon signed rank tests demonstrate that the VVD model has no significant difference with the traditional OAA, OAO, DDAG, and DT methods in term of testing accuracy. However, it improves the classification efficiency with two advantages. First, it reduces the computational complexity for both training and testing by using the least number of classifiers. To be specific, for $2^n$-class problem, the training complexity could be reduced from the maximum $O(c \times 2^n \times n^k)$ to $O(c \times a \times n^k)$, while the testing complexity could also be largely reduced due to a smaller number of kernel evaluations and a simpler post-processing task, which are roughly consistent with those of non-$2^n$-class problem. Second, it eliminates the URs by partitioning the feature space in a more effective manner. Our future research may focus on the problem of optimal class order assignment.

Acknowledgements

This work is partially supported by City University of Hong Kong Research Grant 7002729, and the National Natural Science Foundation of China under the Grants 71171080, 61170107 and 61272289.

References
