計劃成果報告

A Comparison of Methods for Multiclass Support Vector Machines

Chih-Wei Hsu and Chih-Jen Lin 台大資訊系

NSC91-2213-E002-040 林智仁

***Abstract*—Support vector machines (SVMs) were originally designed for binary classification. How to effectively extend it for multiclass classification is still an ongoing research issue. Several methods have been proposed where typically we construct a multiclass classifier by combining several binary classifiers. Some authors also proposed methods that consider all classes at once. As it is computationally more expensive to solve multiclass problems, comparisons of these methods using large-scale problems have not been seriously conducted. Especially for methods solving multiclass SVM in one step, a much larger optimization problem is required so up to now experiments are limited to small data sets. In this paper we give decomposition implementations for two such "all-together" methods. We then compare their performance with three methods based on binary classifications: "one-against-all," "one-against-one," and directed acyclic graph SVM (DAGSVM). Our experiments indicate that the "one-against-one" and DAG methods are more suitable for practical use than the other methods. Results also show that for large problems methods by considering all data at once in general need fewer support vectors.**

***Index Terms*—Decomposition methods, multiclass classification, support vector machines (SVMs).**

I. Introduction

SUPPORT vector machines (SVMs) [6] were originally designed for binary classification. How to effectively extend it for multiclass classification is still an ongoing research issue. Currently there are two types of approaches for multiclass SVM. One is by constructing and combining several binary classifiers while the other is by directly considering all data in one optimization formulation. Up to now there are still no comparisons which cover most of these methods.

The formulation to solve multiclass SVM problems in one step has variables proportional to the number of classes. Therefore, for multiclass SVM methods, either several binary classifiers have to be constructed or a larger optimization problem is needed. Hence in general it is computationally more expensive to solve a multiclass problem than a binary problem with the same number of data. Up to now experiments are limited to small data sets. In this paper we will give a decomposition implementation for two such "all-together" methods: [25], [27] and [7]. We then compare their performance with three methods based on binary classification: "one-against-all," "one-against-one," and DAGSVM [23].

Manuscript received April 9, 2001; revised August 22, 2001. This work was supported in part by the National Science Council of Taiwan under Grant NSC 89-2213-E-002-106.

The authors are with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: cjlin@csie.ntu.edu.tw).

Publisher Item Identifier S 1045-9227(02)01805-2.

Note that it was pointed out in [11] that the primal forms proposed in [25], [27] are also equivalent to those in [3], [11]. Besides methods mentioned above, there are other implementations for multiclass SVM. For example, [14], [19]. However, due to the limit of space here we do not conduct experiments on them. An earlier comparison between one-against-one and one-against-all methods is in [5].

In Section II, we review one-against-all, one-against-one, and directed acyclic graph SVM (DAGSVM) methods which are based on solving several binary classifications. In Section III, we give a brief introduction to the method in [25], [27] which considers all classes at once and show that the decomposition method proposed in [12] can be applied. Another method which also considers all variables together is by Crammer and Singer [7], which will be discussed in Section IV. Numerical experiments are in Section V where we show that "one-against-one" and DAG methods are more suitable for practical use than the other methods. Results also show that for large problems the method proposed in [25], [27] by considering all variables at once generally needs fewer support vectors. Finally, we have some discussions and conclusions in Section VI.

II. One-Against-All, One-Against-One, and DAGSVM Methods

The earliest used implementation for SVM multiclass classification is probably the one-against-all method (for example, [2]). It constructs k SVM models where k is the number of classes. The ith SVM is trained with all of the examples in the ith class with positive labels, and all other examples with negative labels. Thus given l training data $(x_1, y_1), \ldots, (x_l, y_l)$, where $x_i \in R^n, i = 1, \ldots, l$ and $y_i \in \{1, \ldots, k\}$ is the class of x_i, the ith SVM solves the following problem:

$$\begin{aligned} \min_{w^i, b^i, \xi^i} \quad & \frac{1}{2}(w^i)^T w^i + C \sum_{j=1}^{l} \xi_j^i \\ & (w^i)^T \phi(x_j) + b^i \geq 1 - \xi_j^i, \quad \text{if } y_j = i \\ & (w^i)^T \phi(x_j) + b^i \leq -1 + \xi_j^i, \quad \text{if } y_j \neq i \\ & \xi_j^i \geq 0, \quad j = 1, \ldots, l \end{aligned} \tag{1}$$

where the training data x_i are mapped to a higher dimensional space by the function ϕ and C is the penalty parameter.

Minimizing $(1/2)(w^i)^T w^i$ means that we would like to maximize $2/\|w^i\|$, the margin between two groups of data. When data are not linear separable, there is a penalty term $C \sum_{j=1}^{l} \xi_j^i$ which can reduce the number of training errors. The basic concept behind SVM is to search for a balance between the regularization term $(1/2)(w^i)^T w^i$ and the training errors.

After solving (1), there are k decision functions

$$(w^1)^T\phi(x) + b^1$$
$$\vdots$$
$$(w^k)^T\phi(x) + b^k.$$

We say x is in the class which has the largest value of the decision function

$$\text{class of } x \equiv \arg\max_{i=1,\ldots,k}((w^i)^T\phi(x) + b^i). \quad (2)$$

Practically, we solve the dual problem of (1) whose number of variables is the same as the number of data in (1). Hence k l-variable quadratic programming problems are solved.

Another major method is called the one-against-one method. It was introduced in [15], and the first use of this strategy on SVM was in [9], [16]. This method constructs $k(k-1)/2$ classifiers where each one is trained on data from two classes. For training data from the ith and the jth classes, we solve the following binary classification problem:

$$\min_{w^{ij},b^{ij},\xi^{ij}} \quad \frac{1}{2}(w^{ij})^T w^{ij} + C\sum_t \xi_t^{ij}$$
$$(w^{ij})^T\phi(x_t) + b^{ij} \ge 1 - \xi_t^{ij}, \quad \text{if } y_t = i$$
$$(w^{ij})^T\phi(x_t) + b^{ij} \le -1 + \xi_t^{ij}, \quad \text{if } y_t = j$$
$$\xi_t^{ij} \ge 0. \quad (3)$$

There are different methods for doing the future testing after all $k(k-1)/2$ classifiers are constructed. After some tests, we decide to use the following voting strategy suggested in [9]: if $\text{sign}((w^{ij})^T\phi(x)+b^{ij})$ says x is in the ith class, then the vote for the ith class is added by one. Otherwise, the jth is increased by one. Then we predict x is in the class with the largest vote. The voting approach described above is also called the "Max Wins" strategy. In case that two classes have identical votes, thought it may not be a good strategy, now we simply select the one with the smaller index.

Practically we solve the dual of (3) whose number of variables is the same as the number of data in two classes. Hence if in average each class has l/k data points, we have to solve $k(k-1)/2$ quadratic programming problems where each of them has about $2l/k$ variables.

The third algorithm discussed here is the directed acyclic graph SVM (DAGSVM) proposed in [23]. Its training phase is the same as the one-against-one method by solving $k(k-1)/2$ binary SVMs. However, in the testing phase, it uses a rooted binary directed acyclic graph which has $k(k-1)/2$ internal nodes and k leaves. Each node is a binary SVM of ith and jth classes. Given a test sample x, starting at the root node, the binary decision function is evaluated. Then it moves to either left or right depending on the output value. Therefore, we go through a path before reaching a leaf node which indicates the predicted class.

An advantage of using a DAG is that [23] some analysis of generalization can be established. There are still no similar theoretical results for one-against-all and one-against-one methods yet. In addition, its testing time is less than the one-against-one method.

We have implemented all three methods by modifying our SVM software LIBSVM [4].

III. A Method by Considering All Data at Once and a Decomposition Implementation

In [25], [27], an approach for multiclass problems by solving one single optimization problem was proposed. The idea is similar to the one-against-all approach. It constructs k two-class rules where the mth function $w_m^T\phi(x) + b$ separates training vectors of the class m from the other vectors. Hence there are k decision functions but all are obtained by solving one problem. The formulation is as follows:

$$\min_{w,b,\xi} \quad \frac{1}{2}\sum_{m=1}^k w_m^T w_m + C\sum_{i=1}^l \sum_{m\ne y_i} \xi_i^m$$
$$w_{y_i}^T\phi(x_i) + b_{y_i} \ge w_m^T\phi(x_i) + b_m + 2 - \xi_i^m$$
$$\xi_i^m \ge 0, \quad i = 1,\ldots,l, \quad m \in \{1,\ldots,k\}\backslash y_i. \quad (4)$$

Then the decision function is

$$\operatorname*{argmax}_{m=1,\ldots,k}\left(w_m^T\phi(x) + b_m\right)$$

which is the same as (2) of the one-against-all method. Like binary SVM, it is easier to solve the dual problem here. Following [27], the dual formulation of (4) is

$$\min_\alpha \sum_{i,j} \quad \left(\frac{1}{2}c_j^{y_i}A_iA_j - \sum_m \alpha_i^m\alpha_j^{y_i} + \frac{1}{2}\sum_m \alpha_i^m\alpha_j^m\right)K_{i,j} - 2\sum_{i,m}\alpha_i^m\sum_{i=1}^l\alpha_i^m$$
$$= \sum_{i=1}^l c_i^m A_i, \quad m = 1,\ldots,k \quad (5a)$$
$$0 \le \alpha_i^m \le C, \quad \alpha_i^{y_i} = 0 \quad (5b)$$
$$A_i = \sum_{m=1}^k \alpha_i^m, \quad c_j^{y_i} = \begin{cases} 1 & \text{if } y_i = y_j \\ 0 & \text{if } y_i \ne y_j, \end{cases}$$
$$i = 1,\ldots,l, \quad m = 1,\ldots,k \quad (5c)$$

where $K_{i,j} = \phi(x_i)^T\phi(x_j)$. Then

$$w_m = \sum_{i=1}^l (c_i^m A_i - \alpha_i^m)\,\phi(x_i), \quad m = 1,\ldots,k \quad (6)$$

and the decision function is

$$\arg\max_{m=1,\ldots,k}\left(\sum_{i=1}^l (c_i^m A_i - \alpha_i^m)\,K(x_i,x) + b_m\right).$$

Next we explain that (4) is equivalent to two formulations where one is from [11]

$$\min_{w,b,\xi} \quad \frac{1}{2}\sum_{o<m}^k \|w_m - w_o\|^2$$
$$+ C\sum_{i=1}^l \sum_{m\ne y_i} \xi_i^m w_{y_i}^T\phi(x_i) + b_{y_i}$$
$$\ge w_m^T\phi(x_i) + b_m + 2 - \xi_i^m \quad (7a)$$
$$\sum_{m=1}^k w_m = 0 \quad (7b)$$
$$\xi_i^m \ge 0, \quad i = 1,\ldots,l, \quad m \in \{1,\ldots,k\}\backslash y_i$$

and the other is from [3]

$$\min \frac{1}{2}\sum_{o<m}^{k}\|w_m - w_o\|^2 + \sum_{m=1}^{k} w_m^T w_m + C\sum_{i=1}^{l}\sum_{m\neq y_i}\xi_i^m \tag{8}$$

with the same constraints of (4).

For the optimal solution of (4), from (6) and (5c), we have

$$\sum_{m=1}^{k} w_m = \sum_{m=1}^{k}\sum_{i=1}^{l}\left(c_i^m A_i - \alpha_i^m\right)\phi(x_i) = \sum_{i=1}^{l}\left(A_i - \sum_{m=1}^{k}\alpha_i^m\right)\phi(x_i) = 0.$$

Hence, adding (7b) to (4) does not affect the optimal solution set. Then (7b) implies

$$\sum_{o<m}\|w_o - w_m\|^2 = k\sum_{m=1}^{k} w_m^T w_m$$

so (4) and (7) have the same optimal solution set. Similar arguments can prove the relation between (4) and (8) as well. The equivalence of these formulations was first discussed in [11].

Note that (5) has kl variables where l of them are always zero. Hence we can also say it has $(k-1)l$ variables. Unfortunately (5) was considered as an impractical formula due to this huge amount of variables. Hence in [27] only small problems are tested. In the rest of this section we discuss a possible implementation for solving larger problems.

Remember that when solving binary SVM, a main difficulty is on the density of the kernel matrix as in general $K_{i,j}$ is not zero. Thus currently the decomposition method is the major method to solve binary support vector machines [21], [13], [22], [24]. It is an iterative process where in each iteration the index set of variables are separated to two sets B and N, where B is the working set. Then in that iteration variables corresponding to N are fixed while a subproblem on variables corresponding to B is minimized. The size of B and the selection of its contents are both important issues on designing a decomposition method. For example, the sequential minimal optimization (SMO) by Platt [22] considers only two variables in each iteration. Then in each iteration of the decomposition method the sub-problem on two variables can be analytically solved. Hence no optimization software is needed.

However, such working set selections in the binary case may not work here for the dual problem (5). Instead of only one linear constraint in the dual of (1), now in (5a) we have k linear constraints. Thus we can think (5a) as a system of k linear equations with kl variables. Note that the size of the working set is the number of variables we would like to change in one iteration. If it is less than k, then (5a) may be an over-determined linear system with more equations than variables so the solution of the decomposition algorithm cannot be moved. Therefore, in general we have to select more than k variables in the working set of each iteration. However, this is still not an easy task as bounded constraints (5b) have to be considered as well and we would like to have a systematic way which ensures that the selection leads to the decrease of the objective function. Unfortunately, so far we have not found out effective ways of doing it.

Therefore, instead of working on (4), we consider the following problem by adding $\sum_{m=1}^{k} b_m^2$ to the objective function:

$$\begin{aligned}\min_{w,b,\xi}\quad & \frac{1}{2}\sum_{m=1}^{k}\begin{bmatrix} w_m^T & b_m\end{bmatrix}\begin{bmatrix} w_m \\ b_m\end{bmatrix} \\ & + C\sum_{i=1}^{l}\sum_{m\neq y_i}\xi_i^m\begin{bmatrix} w_{y_i}^T & b_{y_i}\end{bmatrix}\begin{bmatrix}\phi(x_i)\\ 1\end{bmatrix} \\ & \geq \begin{bmatrix} w_m^T & b_m\end{bmatrix}\begin{bmatrix}\phi(x_i)\\ 1\end{bmatrix} + 2 - \xi_i^m \\ & \xi_i^m \geq 0, \quad i = 1,\ldots,l, \quad m \in \{1,\ldots,k\}\backslash y_i. \end{aligned} \tag{9}$$

Then in the dual problem k linear constraints are removed

$$\begin{aligned}\min \sum_{i,j}\quad & \left(\frac{1}{2}c_j^{y_i}A_iA_j - \sum_m \alpha_i^m\alpha_j^{y_i} + \frac{1}{2}\sum_m \alpha_i^m\alpha_j^m\right)(K_{i,j}+1) - 2\sum_{i,m}\alpha_i^m \\ & 0 \leq \alpha_i^m \leq C, \quad \alpha_i^{y_i} = 0 \\ & A_i = \sum_{m=1}^{k}\alpha_i^m, \quad c_j^{y_i} = \begin{cases} 1 & \text{if } y_i = y_j \\ 0 & \text{if } y_i \neq y_j, \end{cases} \\ & i = 1,\ldots,l, \quad m = 1,\ldots,k. \end{aligned} \tag{10}$$

The decision function becomes

$$f(x) = \operatorname*{argmax}_{m=1,\ldots,k}\left(\sum_{i=1}^{l}\left(c_i^m A_i - \alpha_i^m\right)\left(K(x_i,x)+1\right)\right).$$

The idea of using bounded formulations for binary classification was first proposed in [10], [18]. For two-class problems, a detailed numerical study showing that the bounded formulation can achieve similar accuracy as the standard SVM is in [12], where the software BSVM was proposed. Without linear constraints, we hope that the problem of selecting the working set in the decomposition method becomes easier. Of course it is not clear yet if the accuracy will be affected as now k b^2 terms are added to the objective function. We will see the results in the experiment section.

For (10), the same decomposition method as in [12] can be applied. We rewrite (10) as the following general form:

$$\begin{aligned}\min_{\alpha} f(\alpha) \quad & = \frac{1}{2}\alpha^T Q\alpha - 2e^T\alpha \\ & 0 \leq \alpha_i^m \leq C, \quad \alpha_i^{y_i} = 0 \\ & i = 1,\ldots,l, \quad m = 1,\ldots,k \end{aligned} \tag{11}$$

where e is a kl by one vector and Q is a kl by kl matrix. Then in each iteration, the subproblem is as follows:

$$\begin{aligned}\min_{\alpha_B}\quad & \frac{1}{2}\alpha_B^T Q_{BB}\alpha_B - \left(2e_B - Q_{BN}\alpha_N^k\right)^T\alpha_B \\ & 0 \leq (\alpha_B)_i \leq C, \quad i = 1,\ldots,q \end{aligned} \tag{12}$$

where $\begin{bmatrix} Q_{BB} & Q_{BN}\\ Q_{NB} & Q_{NN}\end{bmatrix}$ is a permutation of the matrix Q and q is the size of the working set. Note that as $\alpha_i^{y_i} = 0, \forall i$ are fixed variables, we do not select them into the working set.

Assume that α is the solution of the current iteration and q is the size of the working set. We use the following working set selection of BSVM:

1) Let r be the number of free variables at α and calculate

$$v_i^m \equiv \begin{cases} \min(\nabla f(\alpha)_i^m, 0) & \text{if } \alpha_i^m = 0 \\ -|\nabla f(\alpha)_i^m| & \text{if } 0 < \alpha_i^m < C \\ -\max(\nabla f(\alpha)_i^m, 0) & \text{if } \alpha_i^m = C. \end{cases}$$

2) Select indexes of the largest $\min(q/2, r)$ elements in v, where α_i^m is free (i.e., $0 < \alpha_i^m < C$) into B.

Select the $(q - \min(q/2, r))$ smallest elements in v into B.

The main idea of this working set selection is from Zoutendijk's feasible-direction method [28]

$$\min_d \quad \nabla f(\alpha)^T d \quad -1 \le d \le 1$$
$$d_i^m \ge 0, \quad \text{if } \alpha_i^m = 0, \quad d_i^m \le 0, \quad \text{if } \alpha_i^m = C. \quad (13)$$

If d is an optimal solution of (13), then d_i^m must be $+1$ or -1. The vector v defined above actually has $v_i^m = \nabla f(\alpha)_i^m d_i^m$. Thus selecting the smallest elements in v is like selecting components with the best steepest descent direction. This has been used in some early implementations of the decomposition methods (e.g., [13]). However, it was pointed out in [12] that such a selection of the working set may lead to very slow convergence on some difficult problems. With some theoretical and experimental analyzes, in [12] the authors proposed to include some of the largest elements of v whose corresponding α_i^m are free (i.e., $0 < \alpha_i^m < C$). Therefore, if q is the size of the working set, we pick $q/2$ indexes from the smallest elements of v and the other $q/2$ from the largest of v whose corresponding α_i^m are free.

The major difficulty for the implementation lies in calculating elements of Q which are mainly accessed as Q_{BB} and Q_{BN} in (12). As Q in (5) has a very complicated form, it is essential to avoid some possible computational overheads.

More importantly, the form in (10) is not symmetric to indexes i and j. That is

$$\left(\frac{1}{2}c_j^{y_i}A_iA_j - \sum_m \alpha_i^m \alpha_j^{y_i} + \frac{1}{2}\sum_m \alpha_i^m \alpha_j^m\right) K_{i,j} \neq \left(\frac{1}{2}c_i^{y_j}A_jA_i - \sum_m \alpha_j^m \alpha_i^{y_j} + \frac{1}{2}\sum_m \alpha_j^m \alpha_i^m\right) K_{j,i}.$$

This increases the difficulty of writing down the explicit form of Q. As any quadratic formulation can be written as a symmetric form, we must reformulate (10). This will be explained in (15).

Note that as

$$\frac{1}{2}\alpha^T Q \alpha = \sum_{i,j}\left(\frac{1}{2}c_j^{y_i}A_iA_j - \sum_m \alpha_i^m \alpha_j^{y_i} + \frac{1}{2}\sum_m \alpha_i^m \alpha_j^m\right)(K_{i,j}+1) \quad (14a)$$

$$= \frac{1}{2}\sum_{m=1}^{k}\left(\sum_{i=1}^{l}(c_i^m A_i - \alpha_i^m)\begin{bmatrix}\phi(x_i)\\ 1\end{bmatrix}\right)^T \times \left(\sum_{j=1}^{l}(c_j^m A_j - \alpha_j^m)\begin{bmatrix}\phi(x_j)\\ 1\end{bmatrix}\right) \quad (14b)$$

for any nonzero vector $v \in R^{kl} \times R^1$

$$v^T Q v = \sum_{m=1}^{k}\left(\sum_{i=1}^{l}\left(c_i^m \sum_{o=1}^{k} v_i^o - v_i^m\right)\begin{bmatrix}\phi(x_i)\\ 1\end{bmatrix}\right)^T \times \left(\sum_{j=1}^{l}\left(c_j^m \sum_{o=1}^{k} v_j^o - v_j^m\right)\begin{bmatrix}\phi(x_j)\\ 1\end{bmatrix}\right) \ge 0.$$

Hence, Q is positive semidefinite and (10) is a convex optimization problem. Here (14a) to (14b) is by direct calculation. It is easy to see how the first and third terms of α in (14a) are obtained from (14b). For the second term, we have

$$\sum_{m,i} c_i^m A_i \sum_j \alpha_j^m = \sum_{i,j} A_i \alpha_j^{y_i} = \sum_{i,j}\sum_m \alpha_i^m \alpha_j^{y_i}.$$

Next we discuss the explicit form of Q. In particular, we will show what a column of Q is. We consider the vector α as the following form $[\alpha_1^1, \ldots, \alpha_l^1, \ldots, \alpha_1^k, \ldots, \alpha_l^k]^T$. In addition, for $\alpha_1^m, \ldots, \alpha_l^m$, we assume that if $y_i < y_j$, α_i^m is in a position before α_j^m.

In (10), the quadratic terms compose of three groups. First we have

$$c_j^{y_i} A_i A_j = \begin{cases} \left(\sum_{m=1}^k \alpha_i^m\right)\left(\sum_{m=1}^k \alpha_j^m\right) & \text{if } y_i = y_j \\ 0 & \text{if } y_i \neq y_j. \end{cases}$$

Clearly this part forms a symmetric matrix as follows:

$$\begin{bmatrix} \bar{K}_{(1),(1)} & & & & \bar{K}_{(1),(1)} & \\ & \ddots & & \cdots & & \ddots \\ & & \bar{K}_{(k),(k)} & & & & \bar{K}_{(k),(k)} \\ & \vdots & & & & \\ \bar{K}_{(1),(1)} & & & & & \\ & \ddots & & & & \\ & & \bar{K}_{(k),(k)} & & & \end{bmatrix}$$

where $\bar{K}_{(m),(m)}$ includes all elements $\bar{K}_{i,j} \equiv K_{i,j}+1$ with $y_i = y_j = m$. This implies that for the $((s-1)l + j)$th column, row indexes corresponding to $\alpha_{(y_j)}^r, r = 1, \ldots, k$ will have $\bar{K}_{(y_j),j}$ contributed by this part. We use $\alpha_{(y_j)}^r$ to represent $\{\alpha_i^r \mid y_i = y_j\}$ and $\bar{K}_{(y_j),j}$ for $\{\bar{K}_i \mid y_i = y_j\}$.

For the third term, $\sum_{i,j,m} \alpha_i^m \alpha_j^m \phi(x_i)^T \phi(x_j)$, clearly it forms the following symmetric matrix:

$$\begin{bmatrix} \bar{K} & & \\ & \ddots & \\ & & \bar{K} \end{bmatrix}.$$

Similarly we say that at the $((s-1)l+j)$th column, row indexes corresponding to $\alpha_1^s, \ldots, \alpha_l^s$ will include $\bar{K}_{1,j}, \ldots, \bar{K}_{l,j}$.

The most complicated one is the second part $-2\sum_{i,j,m} \alpha_i^m \alpha_j^{y_i} \bar{K}_{i,j}$ as it is not a symmetric form. To make it symmetric we use the property that any quadratic term $xy = (1/2)xy + (1/2)yx$

$$\begin{aligned} & 2\sum_{i,j,m} \alpha_i^m \alpha_j^{y_i} \bar{K}_{i,j} \\ &= \sum_{i,j,m} \left(\alpha_i^m \alpha_j^{y_i} \bar{K}_{i,j} + \alpha_j^{y_i} \alpha_i^m \bar{K}_{i,j}\right) \\ &= \sum_{i,j,m} \alpha_i^m \alpha_j^{y_i} \bar{K}_{i,j} + \sum_{i,j,m} \alpha_i^{y_j} \alpha_j^m \bar{K}_{i,j}. \end{aligned} \quad (15)$$

Hence for the $((s-1)l+j)$th column, elements corresponding to $\alpha_{(s)}^r, r = 1, \ldots, k$ and $\alpha_i^{y_j}, i = 1, \ldots, l$ will have $-\bar{K}_{(s),j}$ and $-\bar{K}_{i,j}, i = 1, \ldots, l$, respectively.

In summary, the $((s-1)l+j)$th column of Q can be obtained as follows:

1) Obtain the column vector $\bar{K}_{i,j}, i = 1, \ldots, l$. Initialize the $((s-1)l+j)$th column as the a kl by one zero vector.
2) For elements corresponding to $\alpha_1^s, \ldots, \alpha_l^s$, add $\bar{K}_{1,j}, \ldots, \bar{K}_{l,j}$ (from the third part)
3) For elements corresponding to $\alpha_1^{y_j}, \ldots, \alpha_l^{y_j}$, minus $\bar{K}_{1,j}, \ldots, \bar{K}_{l,j}$ (from the second part)
4) For elements corresponding to each of $\alpha_{(y_j)}^1, \ldots, \alpha_{(y_j)}^k$, add $\bar{K}_{(y_j),j}$ (from the first part)
5) For elements corresponding to each of $\alpha_{(s)}^1, \ldots, \alpha_{(s)}^k$, minus $\bar{K}_{(s),j}$ (from the second part)

Thus Q in (12) is a dense but not a fully dense matrix. Its number of nonzero elements is about $O(l^2k)$. For practical implementations, we compute and cache $K_{i,j}$ instead of elements in Q. The reduction of the cached matrix from kl to l further improve the training time of this algorithm.

There are different implementations of the above procedure. As the situation may vary for different computational environment, here we do not get into further details.

We have implemented this method as an extension of BSVM and LIBSVM which will be used for the experiments in Section V. In addition, the software is available at http://www.csie.ntu.edu.tw/~cjlin/bsvm.

IV. Method by Crammer and Singer

In [7], Crammer and Singer proposed an approach for multiclass problems by solving a single optimization problem. We will also give a decomposition implementation here. Basically [7] solves the following primal problem:

$$\begin{aligned} \min_{w_m, \xi_i} \quad & \frac{1}{2}\sum_{m=1}^{k} w_m^T w_m + C\sum_{i=1}^{l} \xi_i \\ & w_{y_i}^T \phi(x_i) - w_m^T \phi(x_i) \geq e_i^m - \xi_i, \quad i = 1, \ldots, l \end{aligned} \quad (16)$$

where $e_i^m \equiv 1 - \delta_{y_i,m}$ and

$$\delta_{y_i,m} \equiv \begin{cases} 1 & \text{if } y_i = m \\ 0 & \text{if } y_i \neq m. \end{cases}$$

Then the decision function is

$$\arg \max_{m=1,\ldots,k} w_m^T \phi(x).$$

The main difference from (4) is that (16) uses only l slack variables $\xi_i, i = 1, \ldots, l$. That is, instead of using ξ_i^m as the gap between each two decision planes, here the maximum of k such numbers is considered

$$\xi_i = \left(\max_m \left(w_m^T \phi(x_i) + e_i^m\right) - w_{y_i}^T \phi(x_i)\right)_+$$

where $(\cdot)_+ \equiv \max(\cdot, 0)$. In addition, (16) does not contain coefficients $b_i, i = 1, \ldots, l$. Note that here we do not have to explicitly write down constraints $\xi_i \geq 0$ as when $y_i = m$, $e_i^m = 0$ so (16) becomes

$$0 \geq 0 - \xi_i$$

which is exactly $\xi_i \geq 0$.

The dual problem of (16) is

$$\begin{aligned} \min_{\alpha} \quad & f(\alpha) = \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} K_{i,j} \bar{\alpha}_i^T \bar{\alpha}_j \\ & + \sum_{i=1}^{l} \bar{\alpha}_i^T \bar{e}_i \sum_{m=1}^{k} \alpha_i^m = 0, \quad i = 1, \ldots, l \end{aligned} \quad (17a)$$

$$\begin{aligned} & \alpha_i^m \leq 0, \quad \text{if } y_i \neq m \\ & \alpha_i^m \leq C, \quad \text{if } y_i = m \\ & i = 1, \ldots, l, \quad m = 1, \ldots, k \end{aligned} \quad (17b)$$

where $K_{i,j} \equiv \phi(x_i)^T \phi(x_j)$

$$\bar{\alpha}_i \equiv \left[\alpha_i^1, \ldots, \alpha_i^k\right]^T, \quad \text{and} \quad \bar{e}_i \equiv \left[e_i^1, \ldots, e_i^k\right]^T.$$

Then

$$w_m = \sum_{i=1}^{l} \alpha_i^m \phi(x_i).$$

If we write $\alpha \equiv [\alpha_1^1, \ldots, \alpha_1^k, \ldots, \alpha_l^1, \ldots, \alpha_l^k]^T$ and $e \equiv [e_1^1, \ldots, e_1^k, \ldots, e_l^1, \ldots, e_l^k]^T$, then the dual objective function can be written as

$$\frac{1}{2}\alpha^T (K \otimes I)\alpha + e^T \alpha$$

where I is an k by k identity matrix and $\otimes$ is the Kronecker product. Since K is positive semidefinite, $K \otimes I$, the Hessian of the dual objective function is also positive semidefinite. This is another way to explain that (17) is a convex optimization problem.

The decision function is

$$\arg \max_{m=1,\ldots,k} \sum_{i=1}^{l} \alpha_i^m K(x_i, x).$$

The main difference between linear constraints (5a) and (17a) is that (5a) has k equations while (17a) has l. It is interesting that they come from the Karush–Kuhn–Tucker (KKT) condition on different primal variables. (5a) is due to the unconstrained variables $b_1, \ldots, b_k$ in (4) while (17a) is from the unconstrained variables $\xi_1, \ldots, \xi_l$. In addition, (17a) is much simpler than (5a) as each of its equations involves exactly k variables. In a sense we can say that (17a) contains l independent equations. Because of this advantage, unlike (5a) where we have to remove linear constraints and use (9), here it is easier to directly conduct working set selections for the decomposition method.

In [7] the authors proposed to choose k variables associated with the same x_i in the working set. That is, $\alpha_i^1, \ldots, \alpha_i^k$ are elements of the working set where the selection of the index i will be discussed in (20). Then the subproblem is

$$\begin{aligned} \min \quad & \frac{1}{2} A \bar{\alpha}_i^T \bar{\alpha}_i + B^T \bar{\alpha}_i \sum_{m=1}^{k} \alpha_i^m = 0 \\ & \alpha_i^m \le C_{y_i}^m, \quad m = 1, \ldots, k \end{aligned} \tag{18}$$

where

$$A = K_{i,i} \quad \text{and} \quad B = \bar{e}_i + \sum_{j \ne i} K_{j,i} \bar{\alpha}_j$$

In addition, $\bar{C}_{y_i}^m, m = 1, \ldots, k$ is a k by one vector with all elements zero except that the (y_i)th component is C.

The main reason of this setting is that (18) is a very simple problem. In [7], an $O(k \log k)$ algorithm was proposed for (18) while in [8], a simple iterative approach was used. Here we use the first method. In addition, it is easier to systematically calculate A and B so many complicated derivations in the previous section are avoided.

Note that the gradient of the dual objective function is

$$\begin{bmatrix} \sum_{j=1}^{l} K_{1,j} \alpha_j^1 + e_1^1 \\ \vdots \\ \sum_{j=1}^{l} K_{1,j} \alpha_j^k + e_1^k \\ \sum_{j=1}^{l} K_{2,j} \alpha_j^1 + e_2^1 \\ \vdots \\ \sum_{j=1}^{l} K_{2,j} \alpha_j^k + e_2^k \\ \vdots \end{bmatrix}.$$

Then B, a k by one vector, can be calculated as follows by the information of the gradient

$$\begin{aligned} B_m &= \frac{\partial f(\alpha)}{\partial \alpha_i^m} - K_{i,i} \alpha_i^m \\ &= \frac{\partial f(\alpha)}{\partial \alpha_i^m} - A \alpha_i^m, \quad m = 1, \ldots, k. \end{aligned}$$

Therefore, during iterations it is essential to always keep the gradient updated. This is done after new $\alpha_i^1, \ldots, \alpha_i^k$ are obtained by (18) and $O(kl)$ operations are needed.

Next we discuss the selection of the working set and the stopping condition of the decomposition method. The KKT condition requires that there are $b_1, \ldots, b_l$ and $\lambda_1^1 \ge 0, \ldots, \lambda_l^k \ge 0$ such that for all $i = 1, \ldots, l, m = 1, \ldots, k$,

$$\sum_{j=1}^{l} K_{i,j} \alpha_j^m + e_i^m - b_i = -\lambda_i^m \quad \text{and} \quad \lambda_i^m \left(\bar{C}_{y_i}^m - \alpha_i^m \right) = 0.$$

They are equivalent to that for all $i = 1, \ldots, l, m = 1, \ldots, k$

$$\begin{aligned} \sum_{j=1}^{l} K_{i,j} \alpha_j^m + e_i^m - b_i = 0 \quad & \text{if } \alpha_i^m < \bar{C}_{y_i}^m \\ \le 0 \quad & \text{if } \alpha_i^m = \bar{C}_{y_i}^m. \end{aligned}$$

We can rewrite this as

$$\max_{\alpha_i^m \le \bar{C}_{y_i}^m} \left(\sum_{j=1}^{l} K_{i,j} \alpha_j^m + e_i^m \right) \le b_i \le \min_{\alpha_i^m < \bar{C}_{y_i}^m} \left(\sum_{j=1}^{l} K_{i,j} \alpha_j^m + e_i^m \right). \tag{19}$$

Then during iterations, we select the next working set $\{\alpha_i^1, \ldots, \alpha_i^k\}$ with i from

$$\arg\max_i \left(\max_{\alpha_i^m \le \bar{C}_{y_i}^m} \left(\sum_{j=1}^{l} K_{i,j} \alpha_j^m + e_i^m \right) - \min_{\alpha_i^m < \bar{C}_{y_i}^m} \left(\sum_{j=1}^{l} K_{i,j} \alpha_j^m + e_i^m \right) \right). \tag{20}$$

In other words, among the l k-component groups of variables, we select the one with the largest violation of the KKT condition. For binary SVM, choosing indexes which most violate the KKT condition has been a common strategy (e.g., [4]) though here instead we choose a whole group at once. Then for variables $\{\alpha_i^1, \ldots, \alpha_i^k\}$ which are selected, they do not satisfy the KKT condition of the subproblem (18) so solving (18) will guarantee the strict decrease on the objective function of the dual problem.

Following (19) the stopping criterion can be

$$\max_i \left(\max_{\alpha_i^m \le \bar{C}_{y_i}^m} \left(\sum_{j=1}^{l} K_{i,j} \alpha_j^m + e_i^m \right) - \min_{\alpha_i^m < \bar{C}_{y_i}^m} \left(\sum_{j=1}^{l} K_{i,j} \alpha_j^m + e_i^m \right) \right) < \epsilon \tag{21}$$

where ϵ is the stopping tolerance.

The convergence of the above decomposition method has been proved in [17]. In addition, [17] shows that the limit of

$$\max_i \left(\max_{\alpha_i^m \le \bar{C}_{y_i}^m} \left(\sum_{j=1}^{l} K_{i,j} \alpha_j^m + e_i^m \right) - \min_{\alpha_i^m < \bar{C}_{y_i}^m} \left(\sum_{j=1}^{l} K_{i,j} \alpha_j^m + e_i^m \right) \right)$$