International Journal of Management Science
Vol 16, No 1, May 2010

Implementing a Branch-and-bound Algorithm for Transductive Support Vector Machines*

Chan-Kyoo Park**

Department of Management, Dongguk University, Seoul 100-715, Korea

(Received: December 21, 2009 / Revised: April 5, 2010 / Accepted: April 8, 2010)

ABSTRACT

Semi-supervised learning incorporates unlabeled examples, whose labels are unknown, as well as labeled examples into learning process. Although transductive support vector machine (TSVM), one of semi-supervised learning models, was proposed about a decade ago, its application to large-scaled data has still been limited due to its high computational complexity. Our previous research addressed this limitation by introducing a branch-and-bound algorithm for finding an optimal solution to TSVM.

In this paper, we propose three new techniques to enhance the performance of the branch-and-bound algorithm. The first one tightens min-cut bound, one of two bounding strategies. Another technique exploits a graph-based approximation to a support vector machine problem to avoid the most time-consuming step. The last one tries to fix the labels of unlabeled examples whose labels can be obviously predicted based on labeled examples. Experimental results are presented which demonstrate that the proposed techniques can reduce drastically the number of subproblems and eventually computational time.

Keywords: Semi-supervised Learning, Transductive Support Vector Machine, Optimal Algorithm, Branch-and-bound Algorithm, Min-cut Bound, Network

1. Introduction

Traditionally, learning by examples, one of machine learning paradigms, is divided into two categories according to the availability of labels: unsupervised learning and supervised learning. Unsupervised learning is employed when the labels of examples

* The author was supported by Dongguk University during the sabbatical year (2009~2010).
** Corresponding author, E- mail: parkck@dongguk.edu

are not available. Typical tasks of unsupervised learning include clustering, novelty detection, and dimensionality reduction ([24]). On the other hand, supervised learning assumes that all labels of examples are known. It tries to find a decision function which can explain the relationship between examples and their labels. Furthermore, the decision function is used to predict the label of an example arising in the future. Recently, semi-supervised learning, a new kind of machine learning paradigm positioned between unsupervised and supervised learning, has attracted much attention due to its wide applicability.

Semi-supervised learning can learn from both labeled examples and unlabeled examples to achieve better performance than supervised learning. Usually, it assumes that there are much more unlabeled examples than labeled examples. This gives semi-supervised learning a unique advantage over the traditional learning paradigms because in many domains a large quantity of unlabeled examples are readily available and easy to collect, while the acquisition of labeled data requires additional time, analysis or judgement made by human experts or special devices. For instance, in protein 3D structure prediction, an example represents a sequence of DNA and its label corresponds to a 3D protein folding structure. It can take months of expensive experimental process by crystallographers to identify the 3D structure of a specific protein ([24]).

Although different semi-supervised learning methods rely on different assumptions, one of the basic assumption for semi-supervised learning is cluster assumption, which says that the examples in one dense cluster are very likely to have the same label ([7, 19]). Based on cluster assumption, semi-supervised learning utilizes the domain structure obtained from a multitude of unlabeled examples, which would be impossible only with a few labeled examples. Since the knowledge of domain structure leads to better generalization ability, semi-supervised learning can improve the performance over supervised learning in most applications.

Popular semi-supervised learning models include self-training ([9]), mixture models ([16]), co-training and multi-view learning ([4, 21]), graph-based learning models ([5, 19, 23]), and transductive support vector machines ([2, 20]). Among those models, our focus is on graph-based learning models and transductive support vector machines. First, graph-based models, which employ the concept of cut or Markov random walk, originate from Blum and Chawla [5]'s work. Blum and Chawla [5] constructed maximum flows network based on the similarity between two examples, and utilized min-

cut, which partitions all examples into two groups so that the sum of similarities of arcs between two groups is minimized, to predict the labels of unlabeled examples. Generally, this approach has one critical shortcoming: The sizes of the two groups partitioned by min-cut may be unbalanced. To remedy the shortcoming, Joachims [12] and Yu *et al.* [22] introduced normalized cut, which was originally developed by Shi *et al.* [18] for image segmentation. In addition, Zhu *et al.* [23] introduced harmonic function by relaxing integer constraints related with unknown labels. Finally, Szummer *et al.* [19] modelled semi-supervised learning task using Markov random walk, which produced the results similar to Zhu *et al.* [23]'s work.

The other semi-supervised method we are concerned with is transductive support vector machines (TSVM), which was proposed by Vapnik [20]. TSVM is an extension of support vector machines (SVM), which is used for supervised learning, so that it allows the presence of examples with unknown label. Given training examples, SVM finds an optimal hyperplane which separates them into two groups with maximum margin and minimum training error. The hyperplane will be used to predict the labels of future examples. On the other hand, in transductive setting, all labeled and unlabeled examples are already given before TSVM starts. Like other semi-supervised learning methods, TSVM is based on the belief that exploiting additional information, such as cluster and density, about the whole data will lead to more accurate prediction. In fact, TSVM was shown to have better performance than SVM, especially when a small number of labeled examples and a relatively large number of unlabeled examples are given ([2, 11]). The improved performance of TSVM, however, is obtained at the sacrifice of computational complexity. SVM is formulated into quadratic programming problem, and many efficient techniques, such as decomposition methods, for solving it have been proposed ([6, 8, 15, 17]). On the other hand, TSVM is formulated into mixed integer quadratic problem due to the presence of unlabeled examples, and the computational time required for TSVM increases sharply as the number of unlabeled examples increases. That is why the applications of TSVM have been restricted to small or medium-sized problems.

Existing researches on TSVM can be divided into two approaches. The first approach is to develop approximating models or techniques to solve TSVM problems. Joachims [11] proposed a local search strategy in which the temporary labels of two unlabeled examples were repeatedly exchanged to improve objective function values. Also, he reported that even an approximate solution to TSVM produced more accu-

rate estimation than SVM. Recently, Bie *et al.* [3] showed that TSVM can be reformulated into semidefinite programming by relaxing integer constraints. Unfortunately, the computational complexity of the semidefinite programming model is still too excessive that medium-sized or large-sized problems cannot be solved in a moderate time by general SDP solver.

The other approach is focused on exactly finding an optimal solution to TSVM. The first trial to find an optimal hyperplane of TSVM was carried out by Bennett *et al.* [2], who solved small-sized TSVM problems using a general commercial package for mathematical programming. Since no domain-specialized techniques were developed, the general IP solver was reported to be able to handle no more than a few hundreds of unlabeled examples. Park [14] first utilized the unique structure of TSVM problem and devised problem-specific bounding techniques. In addition, he proposed a branch-and-bound algorithm equipped with two effective bounding strategies, and showed by experimental results that the proposed algorithm could handle more than one thousand of unlabeled examples. Recently, Olivier *et al.* [13] also used a similar branch-and-bound algorithm but solved a slightly modified TSVM which includes a balancing constraint.

Even though the algorithm proposed in our previous work included some problem-specific enhancements to general branch-and-bound algorithm, it still left much room for further development. In this paper, we will introduce some novel techniques to make BBTSVM more practical. The first subject of improvement is lower bounds which have a crucial impact on performance. We are concerned with min-cut bound that is one of the two lower bounds used in the previous work. A min-cut bound (MC bound) is derived from a network associated with each subproblem generated by BBTSVM. Conceptually, min-cut bound is defined to be the minimum sum of capacities across the arcs connecting two nodes in different partitioned groups. Before calculating a min-cut bound, we need to construct a network where the capacity of each arc is assigned under a certain constraint. By analyzing the property of cuts in a network, we will propose how to relax that constraint so that the capacity of each arc can be increased. Since relaxing the constraint will in turn lead to some increment in the sum of capacities, min-cut bound will be tightened. Then, tighter min-cut bound will eventually enable the branch-and-bound algorithm to cut off subproblems earlier.

Second major improvement is early label-fixing technique, which sets the labels

of unlabeled examples to the proper ones as early as possible. At a subproblem generated by BBTSVM, some of examples are labeled and the others remain unlabeled. A reduced SVM is formed only with those labeled examples and its optimal hyperplane can be found by solving it. When an unlabeled example lies well below or above the hyperplane, its label can be easily predicted. Base on the above idea, we will show that an unlabeled example can be fixed to a certain label if it lies far away from the optimal hyperplane obtained with only labeled examples. Once the labels of some unlabeled examples are fixed at a subproblem by early label-fixing technique, the labels remain fixed throughout all child subproblems derived from the subproblem. Therefore, early label-fixing technique will prevent BBTSVM from generating unnecessary subproblems, which will result in significant reduction of computational efforts.

Finally, we discuss how to avoid solving an SVM for all examples, if possible. During its each iteration, BBTSVM tries to find a good feasible solution by solving a full SVM problem which includes all examples. The optimal solution to the full SVM problem is used for updating the best solution to TSVM that BBTSVM has ever found. However, one of the heaviest computational burdens to BBTSVM is caused by solving the full SVM during each iteration. If it is possible to avoid solving the full SVM, it will apparently save computational time. Fortunately, before solving it, we can compute a lower bound to the optimal objective function value of the full SVM and decide whether or not to solve it. If the lower bound turns out to be greater than or equal to the objective function value of the best solution ever found, solving the full SVM does not contribute to improving the current best solution, and thus does not need to be solved. The procedure for obtaining the lower bound is similar to the one used for min-cut bound.

All techniques proposed in this paper are implemented and their effectiveness is shown through computational experiments. According to the experimental results, TSVM makes less prediction errors than ordinary SVM which learns only from labeled examples. This indicates that utilizing not only labeled examples but also unlabeled examples can improve the accuracy of prediction in real-world problems. In addition, the efficiency of avoiding full SVMs and fixing labels of unlabeled examples turn out to be obvious. Especially, early label-fixing technique dramatically reduces the number of subproblems and computational time. Tightening MC bounds itself seems to have a slight impact on saving computational efforts, but it can speed up BBTSVM significantly when combined with early label-fixing technique.

The paper is organized as the following: In the next section, support vector machines and transductive support vector machines are briefly introduced. Then, in section 3, more detailed descriptions about the branch-and-bound algorithm presented in our previous work are given. In section 4, we discuss how to obtain a tighter min-cut bound and how to compute a lower bound to avoid solving full SVMs. In section 5, the basic idea of early label-fixing method will be described and the formal proof is established. Experimental results are given and the effectiveness of the proposed techniques are discussed in section 6. Finally, we make a conclusion and some remarks about future research directions.

2. Support Vector Machines (SVM) and Transductive Support Vector Machines (TSVM)

Given a set of labeled examples, $\{(\mathbf{x}^1, y_1), \cdots, (\mathbf{x}^l, y_l)\}$, where $\mathbf{x}^i \in \mathbb{R}^n$ and $y_i \in \{-1, +1\}$, support vector machines (SVM) finds an optimal hyperplane which separates the examples into two groups with maximum margin and minimum empirical error. The hyperplane, expressed by $\mathbf{w}^T\mathbf{x}+b=0$, forms the *decision boundary* between two groups. Also, the ***decision function*** of an SVM is denoted by $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}+b$, and will be used to predict the label of an unlabeled example arising in the future: For an unlabeled example $\mathbf{x}$, if $f(\mathbf{x}) \geq 0$, the label of $\mathbf{x}$ is predicted as '+1', and otherwise, its label is predicted as '-1.' As illustrated by Figure 1, the optimal hyperplane separates all examples into two groups according to their labels, and the margin, the distance between the two groups, is equal to $2/\|\mathbf{w}\|$. In Figure 1(b), the unlabeled examples lying above the hyperplane are predicted to have label '+1', and the unlabeled examples lying below the hyperplane are predicted to have label '-1.'

To determine $\mathbf{w}$ and b of $f(\mathbf{x})$, SVM solves the following quadratic programming problem:

$$\min \; f_p(\mathbf{w}, b, \xi) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\mathbf{e}^T\xi$$

$$(SP_0): \ s.t. \ y_i(\mathbf{w}^T\mathbf{x}^i + b) \geq 1 - \xi_i, \ i = 1, \cdots, l$$
$$\xi_i \geq 0, \ i = 1, \cdots, l$$

where C is a constant and $\mathbf{e}$ represents the l-dimensional vector whose components are all 1's. The first term of $f_P(\mathbf{w}, b, \xi)$ is the reciprocal of margin, which indicates that SVM pursues a hyperplane with maximum margin. The other term of $f_P(\mathbf{w}, b, \xi)$ represents the sum of empirical errors that are caused by misclassified examples. When a group of examples having label '+1' overlap with another group of examples having label '-1', it might be impossible to exactly separate the examples into two groups according to their labels. In this case, some examples are inevitably misclassified by the decision function, which leads to empirical errors. Hence, SVM seeks to maximize the margin of a hyperplane while minimizing the sum of empirical errors. The balance between margin and empirical errors is controlled by constant C.

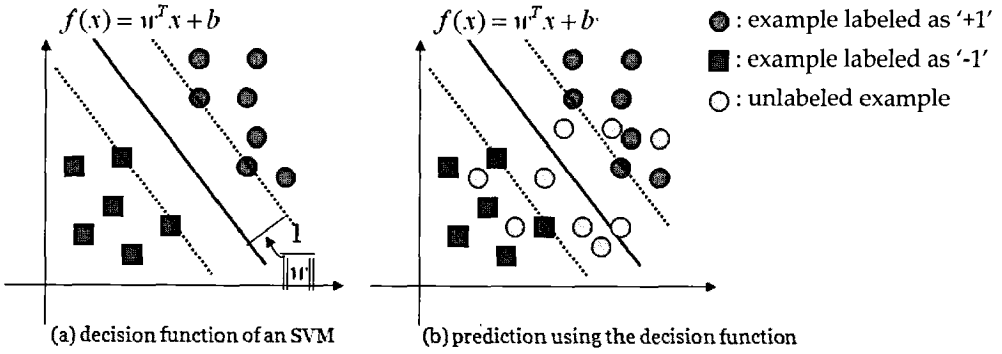


Figure 1. The Decision Function and Prediction of an SVM

In (SP_0), all examples are assumed to be a vector in the n-dimensional space $\mathbb{R}^n$, which is called ***input space***. Usually, these original examples are transformed into the ones in a higher dimensional space, which is called ***feature space***. The mapping of examples from input space to feature space, denoted by $\varphi(\mathbf{x})$, not only increases the possibility that examples can be easily separated, but also enables SVM to incorporate nonlinearity in its decision boundary. In most applications, the mapping results in significant improvement in the SVM's performance. Figure 2 illustrates examples in input space and transformed examples in feature space. As seen in Figure 2(b) and

2(c), the decision function f in input space, which is reversed from f_φ, might include nonlinearity even though the decision function f_φ in feature space is linear.

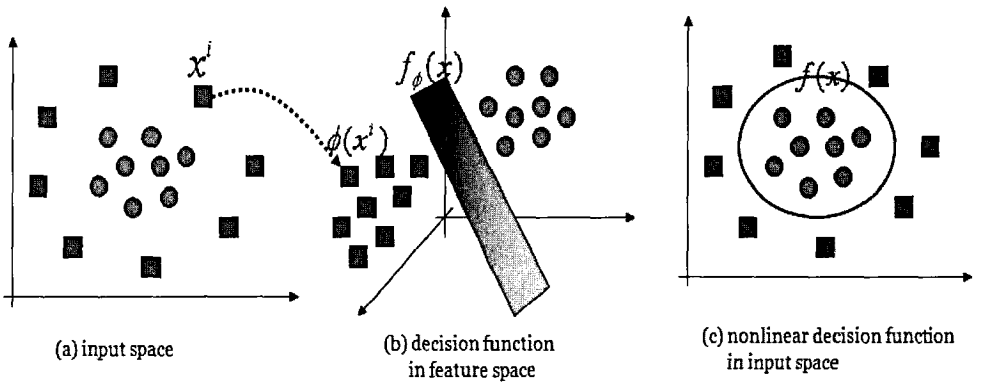


Figure 2. Mapping Examples from Input Space to Feature Space

Fortunately, since all the information that SVM needs to know about the mapping is the value of an inner product between any two examples, the mapping is accomplished in an implicit way using a ***kernel function***. For instance, one of the common kernel functions is radial basis function, which is expressed by

$$k(\mathbf{x}^i, \mathbf{x}^j) = (\varphi(\mathbf{x}^i))^T \varphi(\mathbf{x}^j) = exp(-\frac{\|\mathbf{x}^i - \mathbf{x}^j\|^2}{\rho^2})$$

where ρ is a parameter of radial basis function. The kernel function value, $k(\mathbf{x}^i, \mathbf{x}^j)$, for any pair of two examples is enough for processing the mapping. Furthermore, the normal vector $\mathbf{w}$ to the hyperplane, $\mathbf{w}^T\mathbf{x}+b=0$, can be expressed by a linear combiation of examples like the following ([17]):

$$\mathbf{w} = \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}^i. \tag{1}$$

Using equation (1), the decision function derived from the hyperplane can be rewritten into the following:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}^T \mathbf{x}^i + b. \tag{2}$$

After examples are transformed into feature space, the inner product $\mathbf{x}^T\mathbf{x}^i$ of equation (2) is carried out through the kernel function like the following:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{i=1}^{l} y_i \alpha_i k(\mathbf{x}, \mathbf{x}^i) + b. \tag{3}$$

Thus, SVM including the mapping from input space to feature spaces solves the following transformed problems ([8, 17]):

$$\begin{aligned} &\min \ f_{SP}(\boldsymbol{\alpha}, b, \xi) = \frac{1}{2}\sum_i \sum_j y_i y_j \alpha_i \alpha_j k(\mathbf{x}^i, \mathbf{x}^j) + C\mathbf{e}^T\xi \\ (SP): \ &s.t. \ \ y_i \sum_j (y_j \alpha_j k(\mathbf{x}^i, \mathbf{x}^j) + b) \geq 1 - \xi_i, \ i = 1, \cdots, l \\ &\xi_i \geq 0, \ i = 1, \cdots, l. \end{aligned}$$

Furthermore, the dual problem to (SP) can be constructed like the following ([8, 17]):

$$\begin{aligned} &\max \ f_{SD}(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j k(\mathbf{x}^i, \mathbf{x}^j) \\ (SD): \ &s.t. \ \ \sum_{i=1}^{l} y_i \alpha_i = 0, \\ &0 \leq \alpha_i \leq C, \ i = 1, \cdots, l \end{aligned}$$

Since the constraints of the dual problem have a simple structure which lends itself to the development of efficient algorithms for solving (SD), we hereafter are concerned with the dual problem of SVM rather than its primal problem.

While SVM assumes that the labels of all training examples are already known, transductive support vector machines (TSVM) allows unlabeled examples to be included when finding the decision function. Thus, TSVM determines the labels of unlabeled examples and simultaneously optimal hyperplane so that all examples can

be separated into two groups with maximum margin. Figure 3 illustrates the difference between the decision function of SVM and the decision function of TSVM. SVM constructs its decision function based on only labeled examples, and utilizes it to predict the labels of unlabeled examples. In other words, the construction of the decision function is separated from the prediction and these two tasks are performed sequentially. In contrast, TSVM builds its decision function by considering all examples including unlabeled ones, and determines the labels of unlabeled examples at the same time.

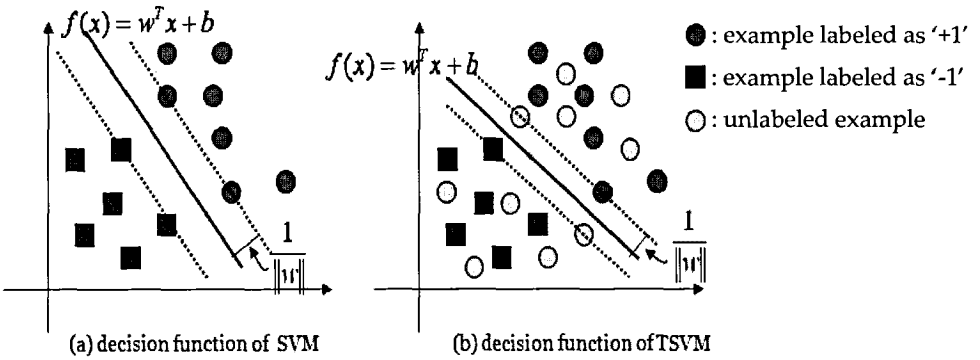


Figure 3. Decision Function of SVM vs. Decision Function of TSVM

Given a set of labeled and unlabeled examples, $\{\mathbf{x}^1, \cdots, \mathbf{x}^l\}$, $\{\mathbf{x}^{l+1}, \cdots, \mathbf{x}^{l+u}\}$, TSVM can be formulated into the following ([20]):

$$\min_{y_{l+1},\cdots,y_{l+u}} \max_{\alpha} f_D(\alpha, y_{l+1}, \cdots, y_{l+u}) = \sum_{i=1}^{l+u} \alpha_i - \frac{1}{2} \sum_{i=1}^{l+u} \sum_{j=1}^{l+u} y_i y_j \alpha_i \alpha_j k(\mathbf{x}^i, \mathbf{x}^j)$$

$$(D): \quad s.t. \quad \sum_{i=1}^{l+u} y_i \alpha_i = 0,$$

$$0 \le \alpha_i \le C, \quad i = 1, \cdots, l+u,$$

$$y_i \in \{-1, 1\}, \quad i = l+1, \cdots, l+u.$$

Note that since y_{l+1} through y_{l+u} represent an integer variable, the objective function of (D) is not quadratic any longer. Also, (D) has integer constraints, thus requiring much more sophisticated approaches for solving it.

3. A branch-and-bound algorithm for TSVM (BBTSVM)

As the first trial to find an optimal solution to (D) by exploiting the unique structure of TSVM, the author proposed a branch-and-bound algorithm for TSVM (BBTSVM) in the previous work. In this section, more detailed descriptions about BBTSVM are presented for further discussions in later sections (For the details, refer to [14]).

Like all kinds of brand-and-bound algorithms, BBTSVM searches subproblems on a branch-and-bound tree to find an optimal solution. Each subproblem on the branch-and-bound tree is generated by selecting one of unlabeled examples, fixing its label to +1 or -1, and changing it into labeled one. So each subproblem can uniquely be identified by the sets of indices of labeled and unlabeled data. For one subproblem, let L^+ and L^- represent the set of indices of examples with label '1' and '-1', respectively, Also, let U denote the set of indices of unlabeled examples. Then, the formulation for the subproblem is written as the following:

$$\min_{\mathbf{y}_U} \max_{\boldsymbol{\alpha}} \; f_D(\boldsymbol{\alpha}, \mathbf{y}_U) = \sum_{i=1}^{l+u} \alpha_i - \frac{1}{2} \sum_{i=1}^{l+u} \sum_{j=1}^{l+u} y_i y_j \alpha_i \alpha_j k(\mathbf{x}^i, \mathbf{x}^j)$$

$$(D(L^+, L^-, U)): \quad s.t. \quad \sum_{i=1}^{l+u} y_i \alpha_i = 0,$$

$$0 \le \alpha_i \le C, \quad i = 1, \cdots, l+u,$$

$$y_i = \begin{cases} +1, & if \;\; i \in L^+ \\ -1, & if \;\; i \in L^- \\ +1 \; or \; -1, & if \;\; i \in U, \end{cases}$$

where $\mathbf{y}_U$ denotes the vector whose each component represents an unknown label. Note that since L^+, L^-, and U are changed for each subproblem, they are different from a corresponding set of indices of originally labeled and unlabeled examples.

During each iteration, BBTSVM selects one subproblem out of the subproblem list. To cut off the subproblem, BBTSVM adopts two bounding strategies: min-cut bound (MC bound) and reduced SVM bound (RSVM bound). If the subproblem is not cut off by the two bounding strategies, BBTSVM solves an SVM problem which consists of all examples including unlabeled examples. After finding a solution to the

full SVM problem, BBTSVM updates the current best solution if the new solution is better. When solving the full SVM problem, all unlabeled examples, whose labels have not yet been fixed, are temporarily assigned a label. Finally, an unlabeled example is chosen so that new subproblems can be generated by branching out from it. The brief outline of BBTSVM is given in Figure 4 where f^* denotes the best objective function value of (D) ever found.

Step 1. Initialize
- $L_0^+ = \{i \mid y_i = +1, i = 1, \cdots, l\}$, $L_0^- = \{i \mid y_i = -1, i = 1, \cdots, l\}$, and $U_0 = \{l+1, \cdots, l+u\}$.
- $f^* = \infty$
- $\mathrm{P} = \{(D(L_0^+, L_0^-, U_0))\}$

Step 2. Select a subproblem $(D(L^+, L^-, U))$ from P

Step 3. Bound the subproblem, if possible
- Compute a min-cut set $[N^+, N^-]$ and the MC bound for $(D(L^+, L^-, U))$.
- If MC bound $\geq f^*$, then cut off the subproblem and go to step 5.
- Compute the RSVM bound for $(D(L^+, L^-, U))$.
- If RSVM bound $\geq f^*$, then cut off the subproblem and go to step 5.
- Solve a full SVM with all labels fixed according to $[N^+, N^-]$, and update f^*, if necessary.

Step 4. Branch
- If $U = \varnothing$, then cut off the subproblem and go to step 5.
- Select an example $\mathbf{x}^i (i \in U)$ to branch out from.
- Add new subproblems $(D(L^+ \cup \{i\}, L^-, U - \{i\}))$ and $(D(L^+, L^- \cup \{i\}, U - \{i\}))$ to P.

Step 5. Repeat step 2 through step 4 until $\mathrm{P} = \varnothing$

Figure 4. A Branch-and-bound Algorithm for TSVM (BBTSVM)

One of the two most important steps of Figure 4 is step 3 where lower bounds for a subproblem are computed. As mentioned before, BBTSVM employs two kinds of lower bound to cut off a subproblem: MC bound and RSVM bound. RSVM bound,

the simpler one of the two, is derived by solving a reduced SVM problem which is made up of only labeled examples for a subproblem. For each subproblem, RSVM bound refers to the optimal objective function value of the reduced SVM problem, which was shown to be less than or equal to the optimal objective function value of $(D(L^+, L^-, U))$ ([14]). On the other hand, MC bound is based on maximum flows problem. For each subproblem, an undirected network is constructed like the following Figure 5:

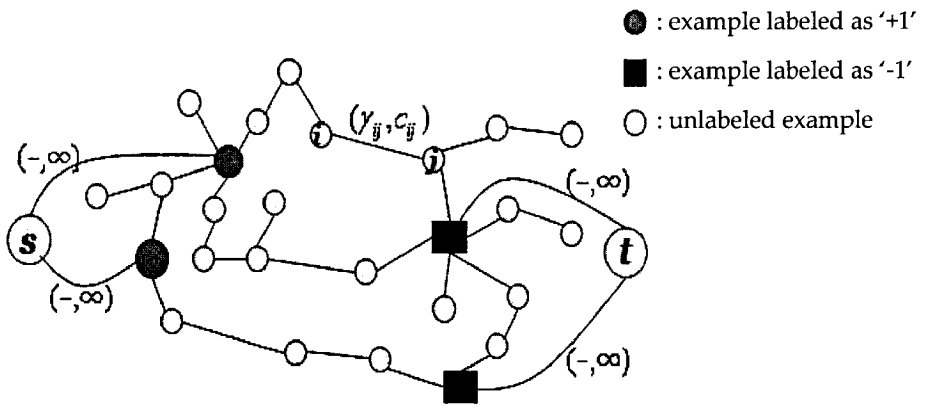


Figure 5. A Network G Constructed for MC Bound

In the network G, each node corresponds to an example. If an example is labeled, the corresponding node is directly connected to the source s or the sink t according to its label. Also, each node is linked with its neighboring nodes by a set of arcs so that the graph becomes connected and includes no more than p paths from the source to the sink.

Two non-negative values, partial multiplier and capacity, are associated with each arch (i, j). First, partial multiplier, denoted by γ_{ij}, is used not only to construct feasible solutions to subproblem $(D(L^+, L^-, U))$ but also to derive the arc's capacity in maximum flows problem. Since the partial multiplier plays a critical role in computing MC bounds, the procedure of how to determine it for each arc is worth detailed review. For each node u, let $J(u)$ denote a subset of nodes adjacent to node u in the network G, and C_u denote a constant modified from C. The partial multiplier, γ_{uj}, for $j \in J(u)$ is assigned so that the following problem is optimized ([14]):

$$(W(J(u), C_u)): \quad \begin{aligned} & max \sum_{j \in J(u)} (2\gamma_{uj} - \frac{p}{2} d_{uj}^2 \gamma_{uj}^2) \\ & s.t. \quad \sum_{j \in J(u)} \gamma_{uj} \leq C_u, \\ & \gamma_{uj} \geq 0, \quad \forall j \in J(u). \end{aligned} \tag{4}$$

Note that the above optimization problem depends on the set of adjacent nodes, $J(u)$, and constant C_u. The optimal value of γ_{uj} to $(W(J(u), C_u))$ can be computed by the following explicit formula([14]):

$$\gamma_{uj}^* = \frac{C_u}{max(C_u, \Delta)} \cdot \frac{2}{pd_{uj}^2} \tag{5}$$

where

$$\Delta = \sum_{j \in J(u)} \frac{2}{pd_{uj}^2}. \tag{6}$$

However, since any arc is incident to two nodes, problem $(W(J(u), C_u))$ cannot be solved independent of its adjacent nodes. To deal with this situation, each node u is chosen in a specific order, and the partial multipliers of arcs incident to u are assigned by solving $(W(J(u), C_u))$. The detailed procedure for determining partial multipliers is presented in the following figure ([14]):

In Figure 6, $Adj(u)$ denotes a set of all nodes adjacent to node u in network G. Note that $J(u) \subset Adj(u)$. In Step 1, for each arc (i, j) which is linked with s or t, its partial multiplier γ_{ij} is set to zero. Step 2 chooses a node u which has not been considered yet. In Step 3, $J(u)$ and C_u are computed. Partial multipliers for some arcs incident to u may have already been fixed if those arcs are also incident to the nodes which were considered before node u. Those arcs are excluded from $J(u)$, and the sum of partial multipliers for those arcs is deducted from C in order to maintain the feasibility of solutions to $(D(L^+, L^-, U))$ which will be constructed from partial multipliers. Step 4 checks whether or not all nodes are considered, and Step 2 is revisited if some nodes have not been considered yet.

Step 1. Initialize

- Set $S = \{s, t\}$ and $\overline{S} = \{1, 2, \cdots, l+u\}$.
- Set $\gamma_{sj} = 0$ for all $j \in Adj(s)$ and $\gamma_{jt} = 0$ for all $j \in Adj(t)$.

Step 2. Select one node u such that $u \in \overline{S}$.

Step 3. Solve $(W(J(u), C_u))$ and set partial multipliers.

- Set $J(u) = Adj(u) - S$.
- Set $C_u = C - \sum_{j \in Adj(u)-J(u)} \gamma_{uj}$.
- Find an optimal solution γ^* to $(W(J(u), C_u))$.
- Set partial multiplier γ_{uj} to γ^*_{uj} for all $j \in J(u)$.

Step 4. Check if all nodes are considered

- Set $S = S \cup \{u\}$ and $\overline{S} = \overline{S} - \{u\}$.
- If $\overline{S} \neq \varnothing$, repeat step 2 through step 4.

Figure 6. A Procedure for Determining Partial Multipliers

Once all partial multipliers are set up, feasible solutions to $(D(L^+, L^-, U))$ can be constructed. Let (N^+, N^-) denote a node partition by which all nodes of G are grouped into two exclusive sets of nodes N^+ and N^-. Throughout the paper, it is assumed that $N^+ \supset L^+$ and $N^- \supset L^-$ for any partition (N^+, N^-). For each node partition (N^+, N^-), let $[N^+, N^-]$ denote the ***cut-set*** which includes all arcs linking one node of N^+ with the other node of N^-. For each cut-set $[N^+, N^-]$, a feasible solution $(\overline{\alpha}, \overline{y}_U)$ to $(D(L^+, L^-, U))$ can be derived by the following equations:

$$\overline{\alpha}_i = \begin{cases} \sum_{j|(i,j)\in[N^+,N^-]} \gamma_{ij}, & \text{if } \{j \mid (i, j) \in [N^+, N^-]\} \neq \varnothing \\ 0, & \text{if } \{j \mid (i, j) \in [N^+, N^-]\} = \varnothing, \end{cases} \tag{7}$$

$$\overline{y}_i = \begin{cases} +1, & \text{if } i \in U \cap N^+ \\ -1, & \text{if } i \in U \cap N^-. \end{cases} \tag{8}$$

In the network G, the other value associated with each arc represents the capa-city in maximum flows problem, which will be denoted by c_{ij}. Each arc's capa-city is

automatically determined from its partial multiplier like the following:

$$c_{ij} = \begin{cases} 2\gamma_{ij} - \frac{p}{2} d_{ij}^2 \gamma_{ij}^2, & \text{if } i \notin \{s, t\} \text{ and } j \notin \{s, t\}, \\ \infty, & \text{if } i \in \{s, t\} \text{ or } j \in \{s, t\}. \end{cases} \tag{9}$$

The capacity of a cut-set is the sum of capacities of all arcs belonging to the cut-set. By solving maximum flows problem, we can obtain a minimum cut-set which minimizes cut-set's capacity across all cut-sets. Park [14] showed that the min-cut, i.e., the capacity of minimum cut-set, is a lower bound to the optimal objective function value of subproblem $(D(L^+, L^-, U))$. More detailed discussions of min-cut bound can be found in [14].

Turning back to BBTSVM in Figure 4, the other step worth review is step 4 where an example to branch out from is selected. To speed up by avoiding unnecessary subproblems, BBTSVM chooses as a branching node the example which is estimated to be the closest to the boundary between two groups of examples. Using the decision function created by a reduced SVM, the extent to which an example is near to the boundary is estimated. The example with minimum absolute decision function value is chosen as a node from which BBTSVM will branch out.

4. Tightening Min-cut Bounds

In this section, we propose two new techniques by which tighter lower bounds to the optimal objective function value of subproblems can be obtained. The first one is for improving MC bound which was originally proposed in our previous work and summarized in the previous section. Getting a tighter bound is crucial because it enables any branch-and-bound algorithm to cut off subproblems earlier and thus reduce the number of subproblems searched. The other one is for computing a lower bound to the optimal objective function value of a full SVM problem which includes all labeled and unlabeled examples. As seen in step 3 of Figure 4, a full SVM problem needs to be solved if the subproblem is not cut off by the two lower bounds. However, solving a full SVM problem is the most time-consuming task in Figure 4. The second technique can prevent BBTSVM from solving a full SVM problem which does not

contribute to finding an improved solution, saving the computational time of BBT SVM significantly.

4.1 Improving an MC bound

As briefly discussed in the previous section, an MC bound is obtained by solving maximum flows problem created for each subproblem. The basic idea for improving MC bounds starts from constraint (4) which requires the sum of γ_{uj}'s to be less than or equal to C_u. If all constraints of $(W(J(u), C_u))$ are ignored, each c_{uj} can be maximized at $\bar{\gamma}_{uj} = 2/(pd_{uj}^2)$. Thus, when $\Delta \leq C_u$, constraint (4) puts no restriction on the value of γ_{uj}, which means that it becomes a non-binding constraint. However, in most cases, Δ exceeds C_u, which indicates that γ_{uj} is restricted by constraint (4), and its optimal value γ_{uj}^* becomes less than $\bar{\gamma}_{uj}$. Therefore, as C_u gets larger, the optimal objective function value of $(W(J(u), C_u)$, or the sum of capacities, is more likely to increase. That is, if constraint (4) can be relaxed so that its right-handed side will have a larger value, then MC bounds can be tightened.

Now we discuss how to increase the right-handed side of constraint (4) without impairing the feasibility of a solution $(\bar{\alpha}, \bar{y}_U)$ which is computed by equations (7) and (8). Let us consider a maximum flows network as in Figure 7 which is constructed for computing an MC bound.

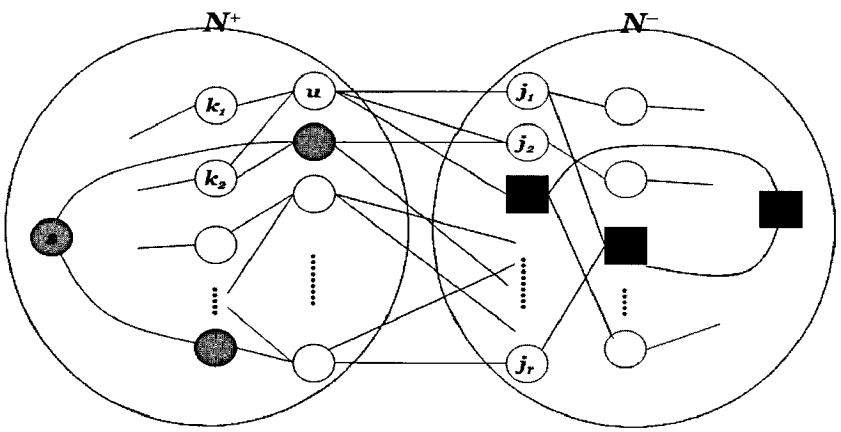


Figure 7. A Cut-set $[N^+, N^-]$ of a Maximum Flows Network

Step 3. Solve $(W(J(u), C_u)$ and set partial multipliers

- Set $J(u) = Adj(u) - S$.
- Set $\hat{C}_u = C - \sum_{j \in Adj(u)-J(u)} \gamma_{uj}$.
- Find an optimal solution $\hat{\gamma}$ to $(W(J(u), \hat{C}_u))$.
- Compute $\gamma_u^{min} = min(min_{j \in Adj(u) \cap S} \gamma_{uj}, min_{j \in J(u)} \hat{\gamma}_{uj})$.
- Set $C_u = \hat{C}_u + \gamma_u^{min}$.
- Find an optimal solution γ^* to $(W(J(u), C_u))$.
- Set γ_{uj} to γ_{uj}^* for all $j \in J(u)$.

Figure 8. The Modified Procedure for Determining Partial Multipliers

Some arcs incident to node u, such as (u, j_1), (u, j_2), and (u, j_3), belong to cut-set $[N^+, N^-]$, but the others, (u, k_1) and (u, k_2) do not belong to cut-set $[N^+, N^-]$. If some arcs incident to node u are contained in a cut-set, then node u always has at least one incident arc which is incident to u but not contained in the cut-set (Note that since no arcs incident to s or t can be contained in minimum cut-set, any cut-set including an arc incident to s or t will be excluded from our consideration). This observation can be applied to any node except s and t. Furthermore, only the arcs contained in the cut set are involved in computing a feasible solution to $(D(L^+, L^-, U))$ by equation (7). In other words, $\bar{\alpha}_u$ computed by equation (7) has nothing with the partial multipliers of the arcs that are not contained in the cut-set. This implies that $\bar{\alpha}_u$ is always less than C by the sum of the partial multipliers of the arcs that do not belong to the cut-set. Therefore, the right-handed side of constraint (4) can be increased by that sum. However, it is not easy to compute the exact amount by which the right-handed side of constraint (4) can be increased. The exact amount depends on a cut-set and ultimately the set of arcs that are not contained in the cut-set. In this paper, we adopt an approximation scheme. The basis idea for the approximation scheme is as the following: For any cut-set, node u has at least one incident arc, (u, v), that does not belong to the cut-set. In addition, the partial multiplier of arc (u, v) is not less than the minimum of the partial multipliers of all arcs incident to u. Hence, the minimum increment, by which the right-handed side of constraint (4) can

be increased for any cut-set, is equal to $\min_{j \in Adj(u)} \gamma_{uj}$. Considering this observation, we can modify step 3 in Figure 6 into the following:

The first three lines of the modified step 3 are the same as those of the original step 3 in Figure 6. Then, the modified step 3 calculates the minimum increment, γ_u^{min}, by which the right-handed side of constraint (4) will be increased. After adding γ_u^{min} to the initial right-handed side value, the modified step 3 finds new partial multipliers. Theorem 1 formally establishes that the partial multipliers obtained by the modified step 3 are also valid for computing a feasible solution to any subproblem.

Theorem 1 *For an arbitrary cut-set* $[N^+, N^-]$ *of subproblem* $(D(L^+, L^-, U))$, *let* $(\bar{\alpha}, \bar{y}_U)$ *be computed by equations (7) and (8) based on partial multipliers,* γ_{ij}, *which are determined by the modified procedure in Figure 8. Then,* $(\bar{\alpha}, \bar{y}_U)$ *is a feasible solution to* $(D(L^+, L^-, U))$.

Proof. By definition, $(\bar{\alpha}, \bar{y}_U)$ satisfies the first and third constraint of $(D(L^+, L^-, D))$. In fact, the first constraint has no relation with the way how partial multipliers are determined. Therefore, it is enough to show that $\bar{\alpha}_u \le C$ for every node u. Let $J(u)$, $\hat{C}_u$, $\hat{\gamma}$, γ_u^{min}, C_u and γ^* be defined as in Figure 8. Suppose that $\bar{\alpha}_u > 0$ (Otherwise, $\bar{\alpha}_u = 0$, which means that the theorem holds trivially). Since $Adj(u) - [N^+, N^-] \neq \varnothing$, the following equation holds:

$$\begin{aligned}
\bar{\alpha}_u &= \sum_{j \in Adj(u) \cap [N^+, N^-]} \gamma_{uj} \\
&= \sum_{j \in (Adj(u) - J(u)) \cap [N^+, N^-]} \gamma_{uj} + \sum_{j \in J(u) \cap [N^+, N^-]} \gamma_{uj} \\
&= \sum_{j \in (Adj(u) - J(u)) \cap [N^+, N^-]} \gamma_{uj} + \sum_{j \in J(u) \cap [N^+, N^-]} \gamma_{uj}^* \\
&\le C + \gamma_u^{min} - min(min_{j \in Adj(u) - J(u)} \gamma_{uj}, \; min_{j \in J(u)} \gamma_{uj}^*)
\end{aligned} \tag{10}$$

Since $C_u \ge \hat{C}_u$, we can see easily from (5) that $\gamma_{uj}^* \ge \hat{\gamma}_{uj}$ for all $j \in J(u)$. Hence, it follows that

$$\gamma_u^{min} = min(min_{j \in Adj(u)-J(u)}\ \gamma_{uj},\ min_{j \in J(u)}\ \hat{\gamma}_{uj})$$
$$\leq min(min_{j \in Adj(u)-J(u)}\ \gamma_{uj},\ min_{j \in J(u)}\ \gamma_{uj}^{*}). \quad (11)$$

Combining inequalities (10) and (11), we can see that $\bar{\alpha}_u \leq C$. □

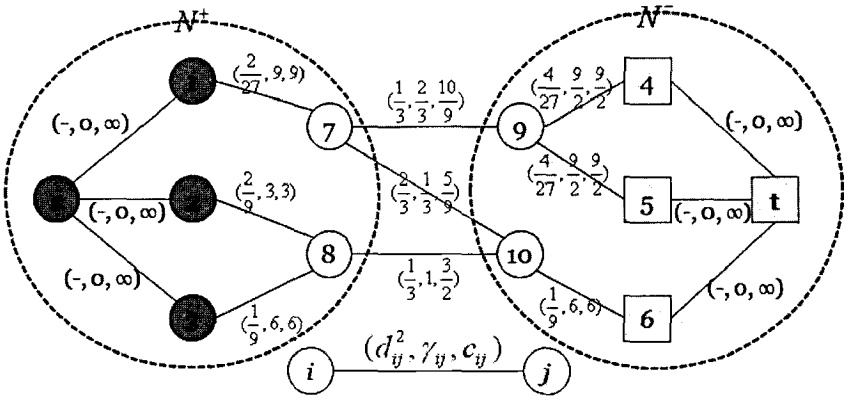
4.2 How to avoid solving a full SVM

In step 3 of Figure 4, if the two lower bounds fail to cut off a subproblem, BBTSVM solves an SVM problem including the entire examples. Since solving this full SVM problem requires the largest portion of computational efforts among all steps of BBTSVM, the performance will be significantly improved if it is possible to avoid solving it. The basic idea of avoiding it is that a lower bound to a full SVM problem can be used to decide whether the full SVM problem needs to be solved or not. If the lower bound is equal to or greater than the best objective function value ever found, the full SVM problem does not have to be solved.

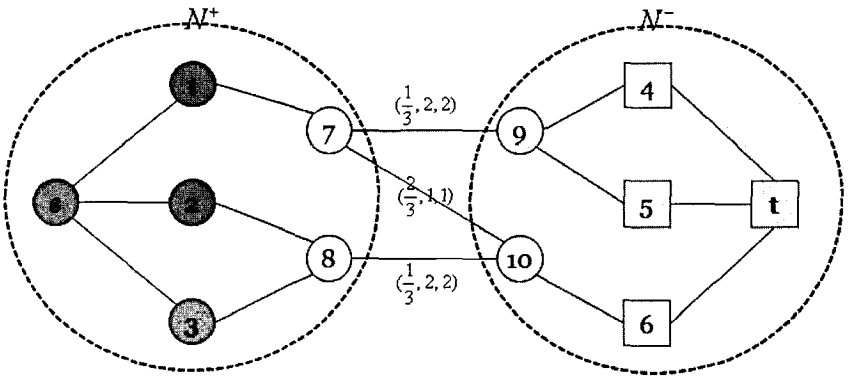
Now we discuss how to obtain a lower bound to this full SVM problem. A similar approach as used for obtaining MC bounds can be applied. The difference is that rather than all arcs in network G, only the arcs in the minimum cut-set are considered for computing a lower bound to a full SVM. As discussed before, the capacity of an arc is determined by its partial multiplier. The procedures presented in Figure 6 and Figure 8 both consider the entire network and assign a value to partial multipliers of all arcs. Moreover, no partial multipliers cannot be set independently of other partial multipliers because they have an impact on each other through constraint (4). Thus, the capacity of each arc in any cut-set is more likely to be maximized by the modified procedure in Figure 8 when only the arcs in the cut-set are considered than when all arcs in the network are considered.

For instance, suppose that a subproblem $(D(L^+, L^-, U))$ is selected by BBTSVM. Also, suppose that the network given in Figure 9(a) is constructed, and f^*, the best objective function value of (D) ever found, is equal to 9/2. Each node was chosen sequentially from node 1 to node 10, and the partial multipliers and capacities of its incident arcs were determined by the procedure in Figure 6. After that, the minimum cut-set of the network was found to be $[N^+, N^-]$ where $N^+ = \{s, 1, 2, 3, 7, 8\}$ and N^+

$= \{t, 4, 5, 6, 9, 10\}$. The MC bound, or the capacity of cut-set $[N^+, N^-]$, is equal to $19/6$, which indicates that it fails to cut off the subproblem. If the RSVM bound is less than 9/2, BBTSVM needs to solve a full SVM including all examples whose labels are fixed according to $[N^+, N^-]$. However, a tighter lower bound can be found by considering only the three arcs in $[N^+, N^-]$ and applying the procedure in Figure 6 to them. The new partial multipliers and capacities are presented in Figure 9(b). Now, the new bound to the full SVM problem is 5, which is greater than f^*. Since the optimal solution to the SVM problem will not improve f^*, the full SVM problem does not need to be solved. Therefore, the computations required for solving it can be saved.



(a) Capacities assigned when all arcs are considered



(b) Capacities assigned when only the arcs in cut-set are considered

Figure 9. Partial Multipliers and Capacities for Different Sets of Arcs Considered

One thing we need to take caution against is that the new bound obtained from Figure 9 should not be used for cutting off subproblem $(D(L^+, L^-, U))$. An MC bound represents a lower bound to all full SVM problems which could be constructed from $(D(L^+, L^-, U))$ by giving an arbitrary label to its unlabeled examples. In con-trast, the new bound obtained by considering only the arcs in the cut-set represents a lower bound to one full SVM problem which is constructed from $(D(L^+, L^-, U))$ by setting the label of its unlabeled examples according to $[N^+, N^-]$. Different labeling of unlabeled examples might lead to a full SVM problem whose objective function value is less than the new bound. Therefore, even though the new bound is greater than f^*, the current subproblem cannot be cut off.

5. Fixing the Label of an Unlabeled Example Early

When branching, BBTSVM creates new subproblems by fixing the labels of unlabeled examples one by one. As more unlabeled examples are changed into labeled one through branching, subproblems become to have some unlabeled examples whose labels can be obviously predicted from their labeled examples. Fixing the labels of those examples will definitely enhance the computational efficiency of BBTSVM in two ways. First, BBTSVM can avoid creating unnecessary subproblems which do not improve the best objective function value ever found. Even though an example is obviously expected to have a certain label, BBTSVM in Figure 4 has no other choice but to create subproblems in which the example is set to the wrong label. Hence, fixing the label of an example to the appropriate one will prevent bbsvm from generating those unnecessary subproblems. The other advantage is that the depth of a leaf node on branch-and-bound tree, i.e., the minimum number of branches made before reaching a leaf node from the root, can be reduced. Since only one unlabeled example is given a label during each branching, the depth of any leaf node is always the same as the number of unlabeled examples of the original TSVM problem. Thus, along the path from the root to a leaf node, BBTSVM generates as many subproblems as the number of unlabeled examples of (D). Moreover, in worst case, BBTSVM needs to compute lower bounds and solve a full problem for each subproblem along the path. If the depth of any leaf node is decreased by fixing together the labels of some

unlabeled examples, the number of subproblems to be searched by BBTSVM will be reduced.

Now we address how to predict and fix the labels of unlabeled examples. The best objective function value f^* plays an important role in fixing them. Let $f(\mathbf{x}) = f(\mathbf{x}) = \bar{\mathbf{w}}^T\mathbf{x} + \bar{b}$ denote the decision function of the reduced SVM which is constructed for a subproblem $(D(L^+, L^-, U))$. Also, let $\mathbf{x}^k$ be one of the unlabeled examples of $(D(L^+, L^-, U))$. If $f(\mathbf{x}^k)$ is sufficiently large, we can expect that $\mathbf{x}^k$ will have label '+1' for all descendant subproblems derived from $(D(L^+, L^-, U))$. The basic idea for the proof is as the following: Suppose that for an unlabeled example $\mathbf{x}^k$, $f(\mathbf{x}^k)$ is sufficiently large and y_k is set to label '-1'. This assumption will cause large empirical errors to a new subproblem $(D(L^+ \cup \{k\}, L^-, U - \{k\}))$ because $\mathbf{x}^k$ is expected to have label '+1'. Thus, the optimal objective function value of $(D(L^+ \cup \{k\}, L^-, U - \{k\}))$ is very likely to increase substantially. If $(D(L^+ \cup \{k\}, L^-, U - \{k\}))$ has the optimal objective function value greater than f^*, then it indicates that $(D(L^+ \cup \{k\}, L^-, U - \{k\}))$ will not contribute to improving the current best solution. Therefore, the label of $\mathbf{x}^k$ can be set to '+1' without impairing the optimality of BBTSVM.

Theorem 2 *Suppose that subproblem* $(D(L^+, L^-, U))$ *is selected by* BBTSVM. *Let* $f(\mathbf{x}) = \bar{\mathbf{w}}^T\mathbf{x} + \bar{b}$ *and* $\boldsymbol{\alpha}^*_{L^+ \cup L^-}$ *denote the decision function and the optimal solution, respectively, of the reduced SVM which is constructed for subproblem* $(D(L^+, L^-, U))$. *For each unlabeled example* $\mathbf{x}^k$, *there exists* η_k *such that if* $|f(\mathbf{x}^k)| > 1 + \eta_k$, y_k *can be set to* $sign(f(\mathbf{x}^k))$ *for all descendant subproblems derived from* $(D(L^+, L^-, U))$ *without loss of the optimality of* BBTSVM.

Proof. Let $L = L^+ \cup L^-$ and $L^k = L \cup \{k\}$. Suppose that y_k is set to $-sign(f(\mathbf{x}^k))$. Also, let us consider the reduced SVM problem $(D(L^k))$ which consists of $\mathbf{x}^k$ and all labe-led examples of $(D(L^+, L^-, U))$.

$$\max\ Z(\boldsymbol{\alpha}_L, \alpha_k) = \mathbf{e}_L^T\boldsymbol{\alpha}_L + \alpha_k - \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}_L \\ \alpha_k\end{bmatrix}^T\begin{bmatrix}Q_{LL} & Q_{Lk} \\ Q_{kL} & Q_{kk}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_L \\ \alpha_k\end{bmatrix}$$

$$(D(L^k)): \quad s.t. \quad \mathbf{y}_L^T \boldsymbol{\alpha}_L + y_k \alpha_k = 0$$
$$0 \le \alpha_i \le C, \ i \in L^k$$

If the optimal objective function value of $(D(L^k))$ is greater than or equal to f^*, then all descendant subproblems, which are derived from $(D(L^+, L^-, U))$ with y_k being set to $-sign(f(\mathbf{x}^k))$, will also have an optimal objective function value greater than or equal to f^*. This implies that those subproblems cannot improve f^*, and so they do not need to be considered for further search by BBTSVM. Consequently, without loss of its optimality, BBTSVM can branch out into only the descendant subproblems with y_k being set to $sign(f(\mathbf{x}^k))$.

Now, to establish our claim it is enough to show that when $|f(\mathbf{x}^k)|$ exceeds a threshold value, the optimal objective function value of $(D(L^k))$ becomes greater than or equal to f^*. Consider a more simplified problem $(SO(L^k))$ which is obtained by fixing each α_i of $(D(L^k))$ to α_i^* except two variables, α_p and α_k:

$$\max \quad W(\alpha_p, \alpha_k) = \alpha_p + \alpha_k + \sum_{i \in \bar{L}} \alpha_i^* - \frac{1}{2} \begin{bmatrix} \alpha_p \\ \alpha_k \\ \alpha_{\bar{L}}^* \end{bmatrix}^T \begin{bmatrix} Q_{pp} & Q_{pk} & Q_{p\bar{L}} \\ Q_{kp} & Q_{kk} & Q_{k\bar{L}} \\ Q_{\bar{L}p} & Q_{\bar{L}k} & Q_{\bar{L}\bar{L}} \end{bmatrix} \begin{bmatrix} \alpha_p \\ \alpha_k \\ \alpha_{\bar{L}}^* \end{bmatrix}$$

$$(SO(L^k)): \quad s.t. \quad y_p \alpha_p + y_k \alpha_k = y_p \alpha_p^*$$
$$0 \le y_p, \ y_k \le C$$

where $p \in L$ such that $0 < \alpha_p^* < C$, and $\bar{L} = L - \{p\}$.

The optimal objective function value of $(SO(L^k))$ becomes a lower bound to that of $(D(L^k))$. Let us compute the optimal solution to $(SO(L^k))$. By rewriting the constraint $y_p \alpha_p + y_k \alpha_k = y_p \alpha^*$ into $\alpha_p = \alpha_p^* - y_p y_k \alpha_k$ and substituting it for α_p, the objective function $W(\alpha_p, \alpha_k)$ can be transformed into $\bar{W}(\alpha_k)$ as the following:

$$\bar{W}(\alpha_k) = (1 - y_p y_k)\alpha_k - \frac{1}{2}(Q_{pp}(\alpha_k^2 - 2\alpha_p^* y_p y_k \alpha_k) + Q_{kk}\alpha_k^2 + 2Q_{pk}(\alpha_p^* \alpha_k - y_p y_k \alpha_k^2)$$

$$
\begin{aligned}
&-2y_p y_k \alpha_k Q_{p\bar{L}} \alpha_{\bar{L}}^* + 2\alpha_k Q_{k\bar{L}} \alpha_{\bar{L}}^*) + \bar{W}(0) \\
&= -\frac{(Q_{pp} + Q_{kk} - 2y_p y_k Q_{pk})\alpha_k^2}{2} + (y_p y_k (\alpha_p^* Q_{pp} + Q_{p\bar{L}} \alpha_{\bar{L}}^* - 1) \\
&+ (1 - \alpha_p^* Q_{pk} - Q_{k\bar{L}} \alpha_{\bar{L}}^*))\alpha_k + \bar{W}(0) \\
&= -\frac{d^2(y_p x_p, y_k x_k)\alpha_k^2}{2} + (y_p y_k (y_p f(x_p) - 1) - (y_k f(x_k) - 1))\alpha_k + \bar{W}(0) \qquad (12)
\end{aligned}
$$

where $\bar{W}(0)$ is defined by

$$\bar{W}(0) = W(\alpha_p^*, 0) = \sum_{i \in L} \alpha_i^* - \frac{1}{2}(\alpha_L^*)^T Q_{LL} \alpha_L^*.$$

To find the maximum of $\bar{W}(\alpha_k)$, we first differentiate it like the following:

$$\frac{d\bar{W}(\alpha_k)}{d\alpha_k} = -d^2(y_p x_p, y_k x_k)\alpha_k + (y_p y_k (y_p f(x_p) - 1) - (y_k f(x_k) - 1)). \qquad (13)$$

By setting the right-hand-side of equation (13) to zero, we can see that $\bar{W}(\alpha_k)$ attains its maximum

$$\bar{W}(\tilde{\alpha}_k) = \frac{(y_p y_k (y_p f(x_p) - 1) - (y_k f(x_k) - 1))^2}{2d^2(y_p x_p, y_k x_k)} + \bar{W}(0) \qquad (14)$$

where

$$\tilde{\alpha}_k = \frac{y_p y_k (y_p f(x_p) - 1) - (y_k f(x_k) - 1)}{d^2(y_p x_p, y_k x_k)} \qquad (15)$$

if all constraints of $(SO(L^k))$ are ignored.

To find the optimal solution to $(SO(L^k))$ with all constraints taken into account, let us consider the following two cases:

i) In case of $y_p = y_k$.

The first constraint of $(SO(L^k))$ becomes $\alpha_p + \alpha_k = \alpha_p^*$. The largest value α_k can take is α_p^*. If $\tilde{\alpha}_k \le \alpha_p^*$, then the optimal objective function value of $(SO(L^k))$ is equal to $\bar{W}(\tilde{\alpha}_k)$. On the other hand, if $\tilde{\alpha}_k > \alpha_p^*$, then $\bar{W}(\alpha_k)$ reaches its maximum at $\alpha_k = \alpha_p^*$. Therefore, the optimal objective function value of $(SO(L^k))$ is written as the following:

$$\bar{W}^* = \begin{cases} \dfrac{(y_p f(\mathbf{x}^p) - y_k f(\mathbf{x}^k))^2}{2d^2(\mathbf{x}^p, \mathbf{x}^k)} + \bar{W}(0), & \text{if } \tilde{\alpha}_k \le \alpha_p^* \\ -\dfrac{d^2(\mathbf{x}^p, \mathbf{x}^k)(\alpha_p^*)^2}{2} + (y_p f(\mathbf{x}^p) - y_k f(\mathbf{x}^k))\alpha_p^* + \bar{W}(0), & \text{if } \tilde{\alpha}_k > \alpha_p^* \end{cases} \tag{16}$$

ii) In case of $y_p \neq y_k$.

The first constraint of $(SO(L^k))$ is rewritten into $\alpha_p - \alpha_k = \alpha^*$. The largest value α_k can take is $C - \alpha_p^*$. If $\tilde{\alpha}_k \le C - \alpha_p^*$, then the optimal objective function value of $(SO(L^k))$ is equal to $\bar{W}(\tilde{\alpha}_k)$. On the other hand, if $\tilde{\alpha}_k > C - \alpha_p^*$, then $\bar{W}(\alpha_k)$ reaches its maximum at $\alpha_k = C - \alpha_p^*$. Therefore, the optimal objective function value of $(SO(L^k))$ is expressed as the following:

$$\bar{W}^* = \begin{cases} \dfrac{\left((1 - y_p f(\mathbf{x}^p)) + (1 - y_k f(\mathbf{x}^k))\right)^2}{2d^2(\mathbf{x}^p, -\mathbf{x}^k)} + \bar{W}(0), & \text{if } \tilde{\alpha}_k \le C - \alpha_p^* \\ -\dfrac{d^2(\mathbf{x}^p, -\mathbf{x}^k)(C - \alpha_p^*)^2}{2} + ((1 - y_p f(\mathbf{x}^p)) & \\ \quad + (1 - y_k f(\mathbf{x}^k)))(C - \alpha_p^*) + \bar{W}(0), & \text{if } \tilde{\alpha}_k > C - \alpha_p^* \end{cases} \tag{17}$$

Now we are ready to prove the claim that if $|f(x_k)| \ge 1 + \eta_k$ for some $p \in L$, then $\bar{W}^* \ge f^*$ where

$$\eta_k = \begin{cases} \eta_k^1, & if\ y_p = y_k\ \ and\ \ \alpha_p^* > 0 \\ \eta_k^2, & if\ y_p = -y_k\ and\ \alpha_p^* \le C \end{cases}, \tag{18}$$

$$\eta_k^1 = max\left(\sqrt{2(f^* - \bar{W}(0))d^2(\mathbf{x}^p, \mathbf{x}^k)}, \frac{d^2(\mathbf{x}^p, \mathbf{x}^k)\alpha_p^*}{2} + \frac{f^* - \bar{W}(0)}{\alpha_p^*} \right) - y_p f(x_p) - 1,$$

$$\eta_k^2 = max\left(\sqrt{2(f^* - \bar{W}(0))d^2(\mathbf{x}^p, -\mathbf{x}^k)}, \frac{d^2(\mathbf{x}^p, -\mathbf{x}^k)(C - \alpha_p^*)}{2} + \frac{f^* - \bar{W}(0)}{C - \alpha_p^*} \right) + y_p f(x_p) - 3.$$

If the claim holds, then the optimal objective function value of $(D(L^k))$ is ob-viously greater than or equal to f^* because $\bar{W}^*$ is a lower bound to the optimal ob-jective function value of $(D(L^k))$. Also, note that equation (18) assumes that $\bar{W}(0) < f^*$. Otherwise, the optimal objective function value of $(D(L^k))$ is trivially greater than or equal to f^*, and then the optimal objective function value of $(D(L^+, L^-, U))$ also becomes greater than or equal to f^*. Hence, if $\bar{W}(0) \ge f^*$, no descendant subproblems derived from $(D(L^+, L^-, U))$ needs to be considered, which means that the theorem is trivially established.

First, suppose that $|f(\mathbf{x}^k)| \ge 1 + \eta_k^1$ for some $p \in L$ such that $y_p = y_k = -sign(f(\mathbf{x}^k))$. Since $y_k f(\mathbf{x}^k) < 0$, the following inequality holds by assumption:

$$\begin{aligned} y_k f(\mathbf{x}^k) &\le -1 - \eta_k^1 \\ &= -max\left(\sqrt{2(f_D^* - \bar{W}(0))d^2(\mathbf{x}^p, \mathbf{x}^k)}, \frac{d^2(\mathbf{x}^p, \mathbf{x}^k)\alpha_p^*}{2} + \frac{f^* - \bar{W}(0)}{\alpha_p^*} \right) + y_p f(\mathbf{x}^p) \end{aligned}$$

$$\Leftrightarrow$$

$$\begin{aligned} & y_p f(\mathbf{x}^p) - y_k f(\mathbf{x}^k) \\ & \ge max\left(\sqrt{2(f^* - \bar{W}(0))d^2(\mathbf{x}^p, \mathbf{x}^k)}, \frac{d^2(\mathbf{x}^p, \mathbf{x}^k)\alpha_p^*}{2} + \frac{f^* - \bar{W}(0)}{\alpha_p^*} \right) \end{aligned} \tag{19}$$

Inequality (19) can be rewritten into the following two inequalities:

$$y_p f(\mathbf{x}^p) - y_k f(\mathbf{x}^k) \geq \sqrt{2(f^* - \bar{W}(0))d^2(\mathbf{x}^p, \mathbf{x}^k)} \tag{20}$$

and

$$y_p f(\mathbf{x}^p) - y_k f(\mathbf{x}^k) \geq \frac{d^2(\mathbf{x}^p, \mathbf{x}^k)\alpha_p^*}{2} + \frac{f^* - \bar{W}(0)}{\alpha_p^*}. \tag{21}$$

Some arithmetic manipulations lead inequality (20) to the following inequality:

$$\frac{(y_p f(\mathbf{x}^p) - y_k f(\mathbf{x}^k))^2}{2d^2(\mathbf{x}^p, \mathbf{x}^k)} + \bar{W}(0) \geq f^*. \tag{22}$$

Also, inequality (21) can be transformed into the following inequality by simple manipulations:

$$-\frac{d^2(\mathbf{x}^p, \mathbf{x}^k)(\alpha_p^*)^2}{2} + (y_p f(\mathbf{x}^p) - y_k f(\mathbf{x}^k))\alpha_p^* + \bar{W}(0) \geq f^*. \tag{23}$$

Both inequalities (22) and (23) indicate that $\bar{W}^*$, the optimal objective function value of $(SO(L^k))$, is greater than or equal to f^*.

Next, suppose that $| f(\mathbf{x}^k) | \geq 1 + \eta_k^2$ for some $p \in L$ such that $y_p = -y_k = sign(f(\mathbf{x}^k))$. Since $y_k f(\mathbf{x}^k) < 0$, the following inequality holds by the assumption:

$$\begin{aligned} y_k f(\mathbf{x}^k) &\leq -1 - \eta_k^2 \\ &= -max\left(\sqrt{2(f^* - \bar{W}(0))d^2(\mathbf{x}^p, -\mathbf{x}^k)},\ \frac{d^2(\mathbf{x}^p, -\mathbf{x}^k)(C - \alpha_p^*)}{2} + \frac{f^* - \bar{W}(0)}{C - \alpha_p^*} \right) \\ &\quad - y_p f(\mathbf{x}^p) + 2 \end{aligned}$$

$$\Leftrightarrow$$

$$\begin{aligned} (1 - y_p f(\mathbf{x}^p)) - (1 - y_k f(\mathbf{x}^k)) \geq max(&\sqrt{2(f^* - \bar{W}(0))d^2(\mathbf{x}^p, -\mathbf{x}^k)}, \\ &\frac{d^2(\mathbf{x}^p, -\mathbf{x}^k)(C - \alpha_p^*)}{2} + \frac{f^* - \bar{W}(0)}{C - \alpha_p^*}) \end{aligned} \tag{24}$$

Inequality (24) can be rewritten into the following two inequalities:

$$(1-y_p f(\mathbf{x}^p))+(1-y_k f(\mathbf{x}^k)) \geq \sqrt{2(f^* - \bar{W}(0))d^2(\mathbf{x}^p, -\mathbf{x}^k)} \tag{25}$$

and

$$(1-y_p f(\mathbf{x}^p))+(1-y_k f(\mathbf{x}^k)) \geq \frac{d^2(\mathbf{x}^p, -\mathbf{x}^k)(C-\alpha_p^*)}{2} + \frac{f^* - \bar{W}(0)}{C-\alpha_p^*} \tag{26}$$

Some simple manipulations lead inequalities (25) and (26) to inequalities (27) and (28), respectively.

$$\frac{((1-y_p f(\mathbf{x}^p))+(1-y_k f(\mathbf{x}^k)))^2}{d^2(\mathbf{x}^p, -\mathbf{x}^k)} + \bar{W}(0) \geq f^* \tag{27}$$

$$-\frac{d^2(\mathbf{x}^p, -\mathbf{x}^k)(C-\alpha_p^*)^2}{2} + ((1-y_p f(\mathbf{x}^p))+(1-y_k f(\mathbf{x}^k)))(C-\alpha_p^*) + \bar{W}(0) \geq f^* \tag{28}$$

Both inequalities (27) and (28), together with equation (17), imply that $\bar{W}^*$, the optimal objective function value of $(SO(L^k))$, is greater than or equal to f^*. □

Finally, early label-fixing technique can be performed at any time after a reduced SVM is solved. In this paper, it will be carried out in the branching step of the modified BBTSVM. Note that all other steps but step 4 in Figure 4 remain the same for the modified BBTSVM. The modified branching step for implementing early label-fixing technique is described as the following:

Modified Step 4. Branch

- If $U = \varnothing$, then cut off the subproblem and go to step 5.
- Find F^+ and F^-, sets of unlabeled examples whose labels can be fixed to +1 and -1, respectively.
- Set $L^+ = L^+ \cup F^+$, $L^- = L^- \cup F^-$, and $U = U - (F^+ \cup F^-)$.
- Select an example $\mathbf{x}^i (i \in U)$ to branch out from.
- Add new subproblems $D(L^+ \cup \{i\}, L^-, U - \{i\})$ and $D(L^+, L^- \cup \{i\}, U - \{i\})$ to P.

Figure 10. The Modified Branching Step for Implementing Early Label-fixing Technique

6. Implementation and experimental results

In this section, the performance of the proposed techniques is discussed through computational experiments. All techniques proposed in this paper are implemented by modifying and enhancing the existing implementation used for the previous work. The main parts of both BBTSVM and modified BBTSVM are implemented in C language. NETDIK and PREFLO are used for constructing and solving maximum flows problems. Also, reduced SVMs and full SVMs are solved using LIBSVM [6].

The implementations are tested for the three data which are widely used in machine learning studies: USPS dataset ([10]), German credit dataset ([1]), Ionosphere dataset ([1]). USPS dataset contains 7,290 training examples of handwritten images which represent one of 10 digits. Each example consists of 256(=16×16) gray-scaled values. To transform it into a binary classification problem, only the examples which represent digit 0 or 1 are extracted from the whole data. The resulting data contain 2,199 examples including 1,194 images of digit 0 and 1,005 images of digit 1. German credit dataset contains 1,000 examples, each of which includes 24 kinds of information related with customer's credit. Each example is labeled as '1' or '2' where '1' and '2' mean good credit and bad credit, respectively. The last dataset, Ionosphere dataset, contains 351 examples that represent radar signals received from the ionosphere. Each example includes 34 attributes and is classified into good or bad return. All dimensions of the three dataset were scaled to (-1, 1).

To set up each dataset for semi-supervised learning, a certain number of examples are randomly chosen and given the original label, and the other examples are made unlabeled. This process is repeated five times so that five different data will be created for the same number of labeled examples. For each dataset, the number of labeled examples ranges from 8 to 28 by fours. RBF kernel function was used for all the three datasets where ρ^2 was set to 100 for USPS and 1 for Ionosphere and German credit dataset. Parameter C for (D) was fixed to 10,000 throughout the three datasets. The experimental results are summarized in Table 1:

The first column of Table 1 represents the name of dataset and the total number of examples contained in a dataset. The second column represents the number of examples randomly chosen as labeled examples. All values in the other columns are obtained by averaging the results of five times' repeated experiments for the same

Table 1. The Experimental Results

Dataset		Original BBTSVM(ORG-BBTSVM)					BBTSVM w/ full SVM avoiding(BBTSVM-AVD)		BBTSVM with full SVM avoiding and tighter MC bound (BBTSVM-MC)				BBTSVM with all proposed techniques (BBTSVM-ALL)			
Name ($l+u$)	l	NSUB	Cut-off MC (RSVM)	Full SVM Solved	Error Rate	Time	Full SVM Avoided (Solved)	Time	NSUB	Cut-off MC (RSVM)	Full SVM Avoided (Solved)	Time	NSUB	Cut-off MC (RSVM)	Full SVM Avoided (Solved)	Time
USPS (2199)	8	2566	27 (1244)	1039	0.00	5155.65	898 (141)	625.65	2518	31 (1225)	903 (137)	619.46	976	282 (170)	376 (58)	229.87
	12	1779	15 (860)	644	0.00	2904.14	489 (155)	459.14	1744	19 (849)	498 (148)	452.36	639	147 (133)	183 (72)	169.54
	16	1542	12 (715)	573	0.00	2623.91	450 (123)	373.91	1511	16 (719)	458 (115)	370.21	725	120 (175)	250 (66)	183.60
	20	1962	14 (940)	759	0.00	3571.73	615 (144)	496.73	1942	19 (952)	631 (135)	487.94	685	263 (62)	230 (55)	159.69
	24	588	10 (262)	190	0.00	806.45	131 (59)	151.45	579	10 (278)	142 (52)	148.48	268	61 (57)	85 (27)	70.44
	28	679	7 (287)	197	0.00	798.24	125 (72)	173.24	665	10 (305)	143 (66)	170.68	342	39 (93)	109 (36)	91.36
Ionosphere (351)	8	39187	1324 (17525)	1305	34.97	1431.80	839 (466)	1180.16	38608	1479 (17792)	849 (462)	1171.06	1132	131 (242)	89 (216)	31.83
	12	28428	80 (13652)	2416	34.53	879.49	517 (1900)	724.46	27871	91 (13790)	522 (1885)	718.89	1768	407 (316)	89 (428)	37.35
	16	102727	21 (27252)	23723	4.57	2372.92	1419 (22304)	1947.10	100221	28 (27597)	1441 (22105)	1914.73	91214	30 (22980)	1432 (22039)	1533.07
	20	47598	25 (22314)	5000	33.96	1351.98	135 (4864)	1311.35	46756	36 (22563)	138 (4807)	1288.42	5998	28 (1297)	37 (1855)	150.14
	24	37916	12 (18089)	5743	33.79	1022.02	174 (5569)	969.82	37429	18 (18272)	177 (5487)	949.61	4213	231 (866)	39 (1330)	103.29
	28	156627	16 (76337)	27366	33.79	4661.09	180 (27185)	4606.99	154680	20 (77303)	182 (24941)	4555.46	7900	3499 (154)	10 (2118)	168.13
German (1000)	8	3348	4 (539)	565	28.78	511.26	8 (557)	459.26	3315	4 (545)	9 (551)	454.72	2267	3 (446)	16 (693)	358.43
	12	19775	17 (8412)	1505	29.68	2284.36	44 (1461)	1998.36	19387	19 (8464)	47 (1454)	1959.17	1763	14 (370)	10 (565)	274.44
	16	3721	4 (819)	543	29.64	549.87	9 (534)	491.37	3613	4 (849)	10 (523)	477.06	1537	34 (250)	14 (466)	218.94
	20	27534	8 (12219)	1448	29.54	2893.65	32 (1416)	2685.65	27127	8 (12359)	34 (1390)	2645.96	1981	5 (430)	10 (706)	309.70
	24	5139	3 (1773)	906	29.40	733.07	21 (885)	596.57	5038	3 (1819)	35 (877)	584.87	2122	770 (53)	34 (585)	208.25
	28	28624	5 (10535)	5104	29.38	3954.03	201 (4903)	2647.53	28341	5 (10863)	213 (4849)	2621.32	4578	90 (1384)	69 (1352)	577.78

number of randomly chosen labeled examples. The 3-rd through 7-th column show the computational performance of BBTSVM originally proposed in [14]. The 3-rd column, NSUB, represents the total number of subproblems generated. The 4-th presents the number of subproblems cut off by MC bound and RSVM bound, respectively. The 5-th column represents the number of full SVMs solved during the execution of BBTSVM. The 6-th column and 7-th column show the error rate and computational time of BBTSVM, respectively. Since the error rate of BBTSVM has nothing to do with the techniques employed for improving its computational time, the other modified branch-and-bound algorithms in Table 1 have the same error rate as the original BBTSVM. The 8-th and 9-th column show the performance of the modified BBTSVM which includes the technique for avoiding full SVMs. The 8-th column represents the number of full SVMs avoided and solved. The 10-th through 13-th column summarize the computational results of the modified BBTSVM which includes the techniques for tightening MC bound and avoiding full SVMs. Finally, the 14-th through 17-th column present the performance of the modified BBTSVM for which all techniques proposed in this paper are implemented including early label-fixing technique.

First, let us compare the accuracy of inductive SVM, SVM-light, and BBTSVM where inductive SVM means that only labeled examples are utilized for learning and unlabeled examples are set aside for evaluating the accuracy of decision function. The same set of data with the same parameter was also applied for inductive SVM and SVM-light for the comparison of error rate. From Figure 11, we can see that in most cases BBTSVM has lower error rate than inductive SVM, which implies that learning from not only labeled but also unlabeled examples leads to improving the accuracy of decision function. Compared to SVM-light, BBTSVM shows much higher accuracy than SVM-light for USPS and German credit dataset. This result indicates that an optimal approach for TSVM can produce much better prediction than an approximating approach, like SVM-light, does. However, in Figure 11(b), Ionosphere dataset illustrates that an optimal approach cannot always make less prediction errors than an approximating one. It seems that the choice between an optimal approach and an approximating one depends on the unique characteristics of a given problem. However, one remarkable advantage of BBTSVM over SVM-light is that BBTSVM shows stable performance as the number of labeled examples or a set of labeled examples vary. In fact, the error rate for BBTSVM remains almost unchanged as the number of

labeled examples increases. This implies that the accuracy of BBTSVM depends on the problem structure, such as the distribution of examples, rather than which examples or how many examples are chosen as labeled one. On the other hand, SVM-light experiences much severe fluctuations in error rate, which indicates that the success of local search strategy adopted by SVM-light depends heavily on which examples and how many examples are selected as labeled one.

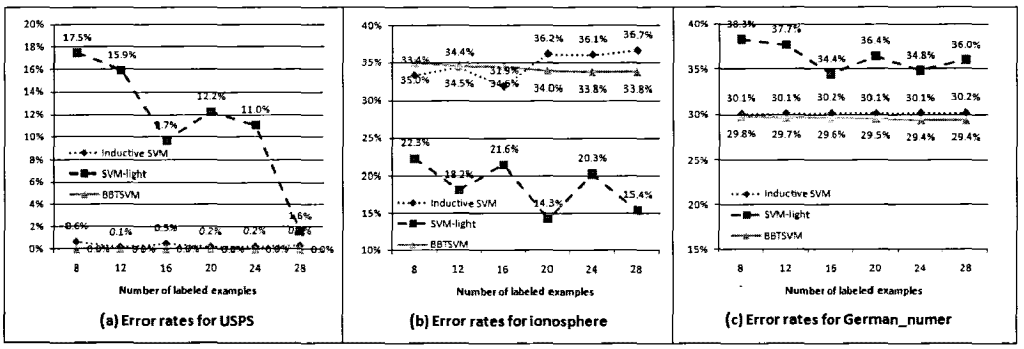


Figure 11. Comparison of Error Rates for the Three Datasets

Now we analyze the efficiency of the proposed techniques in the previous sections. We do not include the CPU times of SVM-light because comparing an optimal approach with an approximating one in terms of CPU time is meaningless. In fact, all problems considered in the experiment were solved within several seconds by SVM-light. Figure 12 summarizes the change in CPU time according to the implemented techniques. First, the efficiency of avoiding full SVMs seems obvious in Figure 12 when comparing the times of ORG-BBTSVM and BBTSVM-AVD. The reduction in CPU time by BBTSVM-AVD is attributed to full SVMs avoided through the technique proposed in section 4. Since the time required for solving full SVMs is mainly determined by the number of examples, more significant reduction in CPU time will be achieved by BBTSVM-AVD for large-sized dataset. Second, tightened MC bound slightly increased the number of subproblems cut off by MC bounds, and then sligh-tly reduced the number of subproblems generated and eventually the CPU times by 1%~2%. Even though the reduction in CPU time by tightened MC bound looks moderate, note that it was still statistically significant. Third, fixing labels of unlabe-led examples led to drastic improvement in the number of generated problems and the CPU time. In many cases, BBTSVM-ALL generated about half or one-third of subproblems

as BBTSVM-MC did, and found an optimal solution roughly twice or three times faster than BBTSVM-MC. In some cases, BBTSVM-ALL was at least eight times faster than BBTSVM-MC. Figure 12 demonstrates that fixing labels of unlabeled examples is a very efficient technique for reducing the number of subproblems and CPU time. Furthermore, early label-fixing has a positive impact on the tightness of MC bounds. Even if BBTSVM-ALL generated less subproblems than BBTSVM-MC, MC bounds in BBTSVM-ALL cut off more subproblems than MC bounds in bbtsvm-MC did. This implies that early label-fixing technique increases the number of labeled examples in the networks where MC bounds are calculated, and eventually makes MC bounds much tighter.

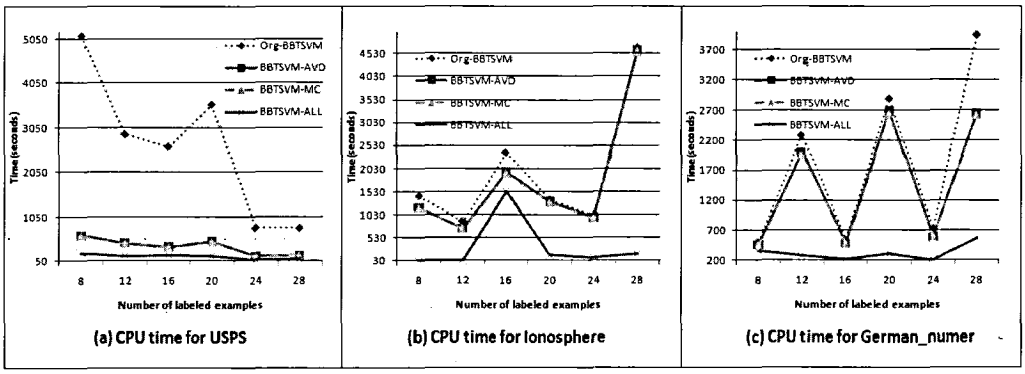


Figure 12. Comparison of CPU Times for the Three Datasets

7. Conclusion

In this paper we proposed the three new techniques which can enhance the computational performance of the branch-and-bound algorithm for transductive support vector machines. The first one, tightening MC bounds, is based on a graph-theoretical observation on cut-set. Since any node in a cut-set has outgoing arcs, the feasibility constraint imposed on partial multipliers can be relaxed so that MC bounds will be more tightened. Even if this technique itself has a moderate impact on saving computational efforts, experimental results showed that it can boost the efficiency of MC bounds when combined with early label-fixing technique. The second technique, avoiding solving full SVMs, finds a lower bound to a full SVM and utilizes it to

prevent BBTSVM from solving unnecessary full SVMs which are most time-consuming. The efficiency of avoiding solving full SVMs was obvious and will be amplified when the number of examples in a full SVM becomes large. The last one, early label-fixing technique, finds unlabeled examples whose label can be definitely predicted, and sets the labels of those examples to the appropriate ones. This technique was proved to drastically reduce the number of subproblems generated by BBTSVM and eventually its computational time.

The paper focused on the computational aspect of an optimal approach for TSVM, but verifying the validity of TSVM for semi-supervised learning is much worth further research. In addition, elaborating min-cut bounds and developing new efficient bounds are still required to make TSVM more practical for large-sized problems in respect of computations.

References

[1] Asuncion, A. and D. J. Newman, UCI Machine Learning Repository (http://www.ics.uci.edu/: mlearn/MLRepository.html), Irvine, CA: University of California, School of Information and Computer Science, 2007.

[2] Bennett, K. P. and A. Demiriz, "Semi-supervised support vector machines," *Advance in Neural Information Processing Systems(NIPS)* 10 (1998), MIT Press.

[3] Bie, T. and De, N. Christianini, "Convex methods for transduction," *Advances in Neural Information Processing Systems(NIPS)* 16 (2004), MIT Press.

[4] Blum, A. and T. Mitchell, "Combining labeled and unlabeled data with co-training," *COLT: Proceedings of the Workshop on Computational Learning Theory* 1998.

[5] Blum, A. and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," *Proceedings of the 18th International Conference on Machine Learning(ICML)* ACM Press, 2001.

[6] Chang, C-C. and C-J. Lin, LIBSVM: a library for support vector machines, http://www.csie.ntu.edu.tw/cjlin/libsvm, 2001.

[7] Chapelle, O., J. Weston, and B. Schölkopf, "Cluster kernels for semi-supervised learning," *Advances in Neural Information Processing Systems (NIPS)* 15 (2003), MIT Press.

[8] Cristianini, J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*, Cambridge University Press, 2000.

[9] Culp, M. and G. Michailidis, "An iterative algorithm for extending learners to a semisupervised setting," *The 2007 Joint Statistical Meeting(JSM)* 2007.

[10] Hull, J. J., "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16 (1994), 550-554.

[11] Joachims, T., "Transductive inference for text classification using support vector machines," *Proceedings of the 20th International Conference on Machine Learning(ICML)* ACM Press, 1999.

[12] Joachims, T., "Transductive learning via spectral graph partitioning," *Proceedings of the 20th International Conference on Machine Learning(ICML)*, ACM Press, 2003.

[13] Olivier, C., V. Sindhwani, and S. S. Keerthi, "Branch and bound for semi-supervised support vector machines," *Advances in Neural Information Processing Systems(NIPS)* 18 (2006), MIT Press.

[14] Park, C-K., "A branch-and-bound algorithm for finding an optimal solution of transductive support vector machines," *Journal of the Korean Operations Research and Management Science Society* 31, 2 (2006), 69-85.

[15] Platt, J., "Fast training of support vector machines using sequential minimal optimization," In: B. Schölkopf, C. J. C. Burges, and A. J. Smola, eds., *Advances in Kernel Methods-Support Vector Learning*, MIT Press, (1999), 85-208.

[16] Ratsaby, J. and S. Venkatesh, "Learning from a mixture of labeled and unlabeled examples with parametric side information," *Proceedings of the Eighth Annual Conference on Computational Learning Theory* (1995), 412-417.

[17] Schölkopf, B. and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2002.

[18] Shi, J. and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000), 888-905.

[19] Szummer, M., T. Jaakkola, "Partially labeled classification with Markov random walks," *Advances in Neural Information Processing Systems(NIPS)* 14 (2002), MIT Press.

[20] Vapnik, V., *Statistical Learning Theory*, Wiley, 1998.

[21] Virginia, R., "Learning classification with unlabeled data," *Advances in Neural Information Processing Systems(NIPS)* 5 (1993), MIT Press.

[22] Yu, S. X. and J. Shi, "Grouping with bias," *Advances in Neural Information Processing Systems(NIPS)* 14 (2001), MIT Press

[23] Zhu, X., Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," *Proceedings of the 20th International Conference on Machine Learning (ICML)*, ACM Press, 2003.

[24] Zhu, X. and A. B. Goldberg, *Introduction to Semi-Supervised Learning*, Morgan and Claypool Publishers, 2009.