TR-IIS-09-002

Tree Decomposition for Large-Scale SVM Problems: Experimental and Theoretical Results

Fu Chang, Chien-Yang Guo, Xiao-Rong Lin, Chi-Jen Lu

April 2, 2009 || Technical Report No. TR-IIS-09-002
http://www.iis.sinica.edu.tw/page/library/LIB/TechReport/tr2009/tr09.html

Tree Decomposition for Large-Scale SVM Problems: Experimental and Theoretical Results

Fu Chang, Chien-Yang Guo, Xiao-Rong Lin, Chi-Jen Lu

Institute of Information Science, Academia Sinica
128 Academia Road, Taipei, 115, Taiwan
{fchang, asdguo, eclipsex527, cjlu}@iis.sinica.edu.tw

Abstract

To handle problems created by large data sets, we propose a method that uses a decision tree to decompose a data space and trains SVMs on the decomposed regions. Although there are other means of decomposing a data space, we show that the decision tree has several merits for large-scale SVM training. First, it can classify some data points by its own means, thereby reducing the cost of SVM training applied to the remaining data points. Second, it is efficient for seeking the parameter values that maximize the validation accuracy, which helps maintain good test accuracy. Third, we can provide a generalization error bound for the classifier derived by the tree decomposition method. For experiment data sets whose size can be handled by current non-linear, or kernel-based SVM training techniques, the proposed method can speed up the training by a factor of thousands, and still achieve comparable test accuracy.

Keywords: binary tree, generalization error bound, margin-based theory, pattern classification, tree decomposition, support vector machine, VC theory

1. Introduction

Support vector machines (SVMs) have proven very effective for solving pattern classification problems (Cortes and Vapnik, 1995; Vapnik, 1995). Because of the growing trend to apply them to various domains of interest, including bioinformatics, computer vision, data mining and knowledge discovery, the size of training data sets continues to grow at a rapid rate. At the same time, there is an ongoing effort to speed up the SVM training. One approach, called the numerical technique in this paper, seeks efficient numerical solutions to the QP optimization problem involved in the SVM training. A well-known solution called sequential minimal optimization (SMO) breaks a large quadratic programming (QP) problem into a series of smallest possible QP problems (Platt, 1999) to reduce the amount of memory required for computation. In terms of speed, SMO has proven superior to similar methods, such as the projected conjugated gradient “chunking” algorithm (Burges, 1998) and Osuna’s algorithm (Osuna et al., 1997). A recent advance in this direction is an online learning method called LASVM (Bordes et al., 2005), which can converge to the QP optimal solution in one pass of examining training samples. The method can be combined with an active selection of training samples to yield faster training, higher accuracy rates, and simpler models. Another advance in this direction is a method called maximum-gain working set selection (Glasmachers and Igle, 2006), which is significantly faster than SMO on large training sets.

A different type of approach, called *data-reduction* in this paper, reduces a large training data set to one or several small data sets. If only one reduced set is obtained, we call the method *single-set reduction* (SSR); and if multiple reduced sets are obtained, we call the method *multiple-set reduction* (MSR). In the latter case, SVM training is conducted on each of the reduced sets and all the SVMs are combined into a final classifier.

We review MSR methods first. Perhaps the simplest MSR method is bagging (Breiman, 1996). It employs a number of down-sampled data sets to train SVMs, which jointly classify a test object based on majority vote. The boosting method (Schapire, 1990) trains SVMs in a sequential manner, and the training of a particular SVM is dependent on the training and performance of previously trained SVMs. The divide-and-combine strategy (Rida et al., 1999) decomposes an input space into possibly overlapping regions, assigns each region a local predictor, and combines the local predictors into a global solution to the prediction problem. The Bayesian committee machine (Tresp, 2000) partitions a large data set into smaller ones. The SVMs trained on the reduced sets jointly define the posteriori probabilities of the classes into which test objects are categorized. The method proposed by Collobert et al. (2002) divides a set of input samples into smaller subsets, assigns each subset a local expert, and conducts a loop to re-assign samples to local experts according to how well the experts perform. The cascade SVM method (Graf et al.,

2005) also splits a large data set into smaller ones and extracts support vectors (SVs) from each of them. The resulting SVs are further combined and filtered in a cascade of SVMs. A few passes through the cascade ensures that the optimal solution is found.

On the SSR side of the data-reduction approach, the squashing method (Pavlov et al., 2000) uses a likelihood-based squashing technique to obtain a reduced data set, and then trains linear SVMs on that set. The sparse greedy approximation method (Smola and Schölkopf, 2000) constructs a compressed representation of the design matrix involved in the QP problem; while information vector machines (Lawrence et al., 2003) use a sparse Gaussian process to select training samples with criteria based on information-theoretic principles. Clustering-based SVM (Yu et al., 2005) applies a hierarchical clustering algorithm to obtain a reduced data set, which is used to train SVMs. The concept boundary detection (CBD) method (Panda et al., 2006) prepares nearest-neighbor lists as training samples, and uses a special down-sampling technique to extract the data points that lie close to class boundaries. This method can find a single set of near-boundary points for all class pairs. In contrast, many other methods that utilize SVMs to analyze training samples have to find different reduced sets for different class pairs, since SVMs can only work on one class pair at a time. For more details of data-reduction methods proposed up to 2001, readers may refer to Tresp (2001).

Finally, we remark that the numerical and data reduction approaches, instead of competing, can actually complement each other's functions. The data reduction approach must train SVMs on reduced data sets and it can certainly use an efficient numerical method to perform the task. The numerical approach, on the other hand, can benefit by using an efficient data reduction method to reduce its computational burden.

In this paper, we propose a method that decomposes a large data set into a number of smaller ones and trains SVMs on each of them. This approach reduces the total training time for a very simple reason. The time complexity of training an SVM is in the order of n^2 when the number of training samples is n. If each smaller problem deals with σ samples, where $\sigma < n$, then the complexity of solving all the problems is in the order of $(n/\sigma)\times\sigma^2 = n\sigma$, which is much smaller than n^2 if n is significantly larger than σ. Decomposing a large problem into smaller problems has the added benefit of reducing the number of support vectors (SVs) in the resultant SVMs. This in turn reduces the time required for the testing process in which the number of SVs dominates the complexity of the computation.

Our method can be categorized as an MSR method. It differs from other MSR methods in that it uses of a *decision tree* to obtain multiple reduced data sets, whereas other methods use non-supervised clustering (Rida et al., 1999), random sampling (Breiman, 1996), or random partition (Tresp, 2000; Collobert et al., 2002; Graf et al., 2005). Since our method uses a decision tree to decompose the data space, we refer to it as the *tree decomposition* (TD) method and the resultant classifier as the *TD classifier*.

A decision tree decomposes a data space into high-dimensional rectangles. Although the generalization power of the decision tree as a classifier is compromised by the strict requirement of rectangular partition of the data space, the role of the decision tree as a decomposition scheme can have the following benefits when dealing with large-scale SVM problems.

First, the decision tree may decompose the data space so that certain decomposed regions become homogeneous; that is, they contain samples of the same labels. In the testing phase, when a data point flows to a homogeneous region, we simply classify it in terms of the common label of that region. This helps alleviate the burden of SVM training, which is only conducted in heterogeneous regions. In fact, our experiments revealed that, for certain data sets, more than 90% of the training samples reside in homogeneous regions; thus, the decision tree method saves an enormous amount of time when training SVMs. Random partitioning, on the other hand, cannot produce such an effect, since random pooling of a set of samples can hardly create a homogeneous data set due to the independent sampling operation.

Another benefit of using the decision tree is the convenience it provides when searching for all the relevant parameter values to maximize the solution's validation accuracy, which helps maintain good test accuracy. The goal of the TD method is to attain comparable validation accuracy while consuming less time than training SVMs on the full data sets. To achieve our purpose, we found that it is important to control the size σ of the tree-decomposed regions as well as the SVM-parameter values. For some data sets, σ could be set to 1,500, while for other data sets, it had to be set to a larger value. Thus, using tree decomposition for SVMs makes σ an additional parameter to the usual SVM-parameters. Other MSR methods do not attempt to search for the optimal size of decomposed regions. Such searches are particularly easy under the TD method because a decision tree is constructed in a recursive manner; hence, obtaining a tree with a larger size of σ does not require a new training.

Using a decision tree also helps alleviate the cost of searching for the optimal values of SVM-parameters. Searching for these values is important, but it takes a tremendous amount of time, especially when training non-linear SVMs. To the best of our knowledge, no data-reduction method has attempted to reduce the cost of this operation. Our strategy involves training SVMs with all combinations of SVM-parameter values *only* for decomposed regions with the minimum σ-level. The optimal values of the SVM-parameters obtained at this level are not necessarily the same as those obtained at higher levels. However, we observe that the best values for a higher level are usually among the top-ranked values for the minimum level. Therefore, when we want to train SVMs for a higher σ-level, we only train them with the top-ranked values obtained for the minimum level. Given the $O(n^2)$-complexity of SVM training, conducting a full search of SVM-parameter values only in regions with the minimum σ-level certainly reduces the SVM training time. In fact, our experiments showed that such savings were possible even when the optimal σ-level was no less than the full size of the data set.

Although the decision tree method may not be the only way to achieve the above benefits for large-scale SVM problems, its effect can be understood in theory and a generalization error bound can be derived for the TD classifier. The bound is the sum of two terms: the first term dominates in magnitude and is associated with SVM training; and the second term is associated with tree training. Our experiment results show that the numerical value of the dominant term is as small as, or of the same order of magnitude as, its counterpart in a generalization error bound for SVM training conducted on the whole data set. This finding constitutes indirect evidence of the efficacy of tree decomposition for large-scale SVM problems.

Finally, we remark that it is possible to have multiple decompositions of the same data space with multiple trees. These trees can be obtained by using a randomized, rather than the optimal, split point at each tree node (Dietterich, 2000). By so doing, we train

SVMs on all the decomposed regions and classify the test data based on majority votes. We have actually studied the effect of such multiple decompositions. In terms of test accuracy, multiple decompositions are not as effective as searching for the optimal σ-level of decomposed regions. In fact, when the latter search is conducted, introducing multiple decompositions does not lead to any significant improvement. Therefore, to avoid unnecessary complications, in this paper, we only consider the decomposition of a data space by a single decision tree.

In the experimental study, we divided our data sets into training, validation, and test components. We then used the training component to build TD classifiers, the validation component to determine the optimal parameters, and the test component to measure the test accuracy. We adopted two types of SVM training: one-against-one (1A1) (Knerr et al. 1990; Platt et al. 2000) and one-against-others (1AO) (Bottou et al., 1994). Furthermore, we built non-linear SVMs on the data sets. When evaluating the TD method, we found it could train TD classifiers that achieve comparable test accuracy rates to those of SVM classifiers. The speedup factor for the six datasets, whose sizes ranged from 10K to 494K, was between approximately 4 and 3,691 for 1A1 trainings, and between approximately 29 and 5,775 for 1AO trainings. Furthermore, we found that TD achieved much higher speedup factors than two alternative methods, namely, bagging (Breiman, 1996), an MSR method; and CBD (Panda et al., 2006), an SSR method. To demonstrate that TD can efficiently train classifiers for larger data sets, we applied it to two datasets whose sizes were approximately 581K and 4,898K respectively. The first data set took 4.7 and 7.3 hours to complete 1A1 and 1AO trainings respectively, while the second data set required 5.2 hours for both types of training.

The remainder of this paper is organized as follows. In Section 2, we describe the TD method. Section 3 details the experiment results. In Section 4, we provide theoretical results for the TD method. Then, in Section 5, we present some concluding remarks.

2. The TD Method

In this section, we consider the decision tree that we use as the decomposition scheme, and discuss the training process for the TD method. An implementation of the TD method is available at

http://ocrlnx03.iis.sinica.edu.tw/~dar/Download%20area/tdsvm.php3

2.1 The Decision Tree

For the decomposition scheme, we adopt CART (Breiman et al., 1984) or a binary C4.5 scheme (Quinlan, 1986) that allows two child nodes to grow from each node that is not a leaf. Using a C4.5 scheme that allows multiple child nodes is feasible; however, we do not consider it in this paper, since a binary C4.5 performs the job rather well for us.

To grow a binary tree, we follow a recursive process, whereby each training sample flowing to a node is sent to its left-hand or right-hand child node. At a given node E, a certain feature f_E of the training samples flowing to E is compared with a certain value v_E so that all samples with $f_E < v_E$ are sent to the left-hand child node, and the remaining samples are sent to the right-hand child node. The values of f_E and v_E are determined as follows. The *split point* v_f of each feature f is calculated by

$$v_f = \arg\max_v IR(f, v), \tag{1}$$

where

$$IR(f, v) = I(S) - \frac{|S_{f<v}|}{|S|} I(S_{f<v}) - \frac{|S_{f\geq v}|}{|S|} I(S_{f\geq v}),$$

S is the set of all samples flowing to E; $S_{f<v}$ consists of the elements of S with $f < v$; $S_{f\geqq v}$ = $S \backslash S_{f<v}$; $|X|$ is the size of any data set X; and $I(X)$ is the impurity of X. The impurity function we use in our experiments is the entropy measure, defined as

$$I(\mathrm{S}) = -\sum_y p(S_y) \log p(S_y),$$

where $p(S_y)$ is the proportion of S's samples whose label is y. Then,

$$f_E = \arg\max_f IR(f, v_f),$$

and v_E is taken as the split point of f_E.

We stop splitting a node E when one of the following conditions is met: (i) the number of samples that flow to E is smaller than a *ceiling size* σ; or (ii) when $IR(f, v) = 0$ for all f and v at E. The value of σ in the first condition is determined in a data-driven fashion, which we describe in Section 2.2. The second condition occurs when all the samples that flow to E are homogeneous or when a subset of them is homogeneous and the remaining samples, although carrying different labels, are identical to some members of the homogeneous subset. There are other possible cases for the second condition, but their occurrence is extremely rare. If we want to split E in these cases, for a given feature f, we can choose the following split point to minimize the difference between $| S_{f \geqq v} |$ and $| S_{f < v} |$, i.e.,

$$v_f = \arg\min_v | \, | S_{f \le v} | - | S_{f > v} | \, | ,$$

and then choose the feature whose split point has the minimum difference among all features.

After growing a tree, we train an SVM on each of its leaves, using samples that flow to each leaf as training data (Figure 1). The values of the SVM parameters are also determined in a data-driven fashion. A tree and all SVMs associated with its leaves constitute a TD classifier, as shown in Figure 1. In the training phase, all the SVMs are trained with the same parameter values. We describe the determination of the optimal values in Section 2.2. In the validation/testing procedure, we first input a given validation/test object **x** to the tree. If **x** reaches a leaf that contains homogeneous samples, we classify **x** as the label of those samples; otherwise, we classify it with the SVM associated with that leaf.

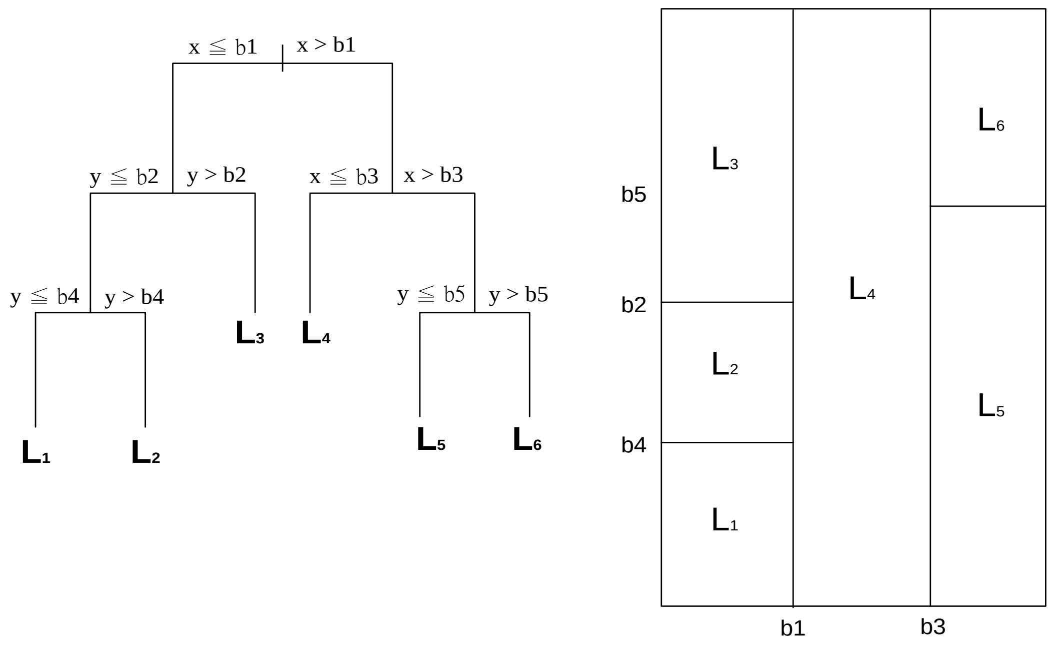

Figure 1. The architecture of a TD classifier: a tree and all its leaves (L_1 to L_6) are produced and SVMs are trained on the leaves.

2.2 The TD Training Process

Given a training and validation component, we build a TD classifier on the training component and determine its optimal parameter values with the help of the validation component. The parameters associated with a TD classifier are: (i) σ, the ceiling size of the decision tree; and (ii) the SVM parameters. Their optimal values are determined in the following way.

In the initial stage, we train a binary tree with an *initial* ceiling size σ_0, and then train SVMs on the tree's leaves with SVM-parameters $\boldsymbol{\theta}$. Note that we express $\boldsymbol{\theta}$ in boldface to indicate that it may consist of more than one parameter. Let $v(\sigma_0, \boldsymbol{\theta})$ be the validation accuracy rate of the resultant TD classifier. Then, we can define

$$\boldsymbol{\theta}_0 = \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta} v(\sigma_0, \boldsymbol{\theta}), \text{ and}$$

$$r(0) = v(\sigma_0, \boldsymbol{\theta}_0),$$

where Θ is the set of all possible SVM-parameter values whose effects we want to evaluate. The value $r(0)$ is the best validation accuracy rate that we obtain out of all the TD classifiers with ceiling size σ_0. In our experiments, we set $\sigma_0 = 1{,}500$.

For later stages, we want to construct TD classifiers with a larger ceiling size, but we only train their associated SVMs with k top-ranked $\boldsymbol{\theta}$. To do so, we rank $\boldsymbol{\theta}$ in the descendant order of $v(\sigma_0, \boldsymbol{\theta})$. Let $\Theta_{[k]}$ be the set that consists of k top-ranked $\boldsymbol{\theta}$. In our experiments, we set k to 5.

To be more specific, at stage t, we set $\sigma_t = 4\sigma_{t-1}$ for $t = 1, 2, \ldots$. We modify the tree with ceiling size σ_{t-1} by dropping a few nodes from the lower levels so that the tree's ceiling size becomes σ_t. We then train new TD classifiers on the modified tree with $\boldsymbol{\theta}$ being chosen from $\Theta_{[k]}$. Then, we define

$$\boldsymbol{\theta}_t = \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta_{[k]}} v(\sigma_t, \boldsymbol{\theta}), \text{ and}$$

$$r(t) = v(\sigma_t, \boldsymbol{\theta}_t).$$

The value $r(t)$ is the best validation accuracy rate out of all the TD classifiers with ceiling size σ_t. We terminate the process when the improvement in the best validation accuracy rate is insignificant, or we have already reached the root node of the tree.

The steps of the TD training process are as follows.

1. Set $t = 0$. Train the TD classifiers with ceiling size σ_0 and choose the SVM-parameters $\boldsymbol{\theta}$ from Θ; then compute $r(0)$.
2. Increase t by 1 and set $\sigma_t = 4\sigma_{t-1}$. Obtain the binary tree with ceiling size σ_t and train the TD classifiers on that tree, and choose the SVM-parameters $\boldsymbol{\theta}$ from $\Theta_{[k]}$; then compute $r(t)$.
3. If $r(t) - r(t-1) < 0.5\%$, or σ_t is no less than the size of the training component, terminate the process; otherwise, proceed to step 2. Note that when the process is terminated, it outputs a TD classifier with the ceiling size σ_{opt} and SVM-parameters $\boldsymbol{\theta}_{opt}$, where $\sigma_{opt} = \sigma_{t-1}$ and $\boldsymbol{\theta}_{opt} = \boldsymbol{\theta}_{t-1}$ if $r(t) - r(t-1) < 0.5\%$, or $\sigma_{opt} = \sigma_t$ and $\boldsymbol{\theta}_{opt} = \boldsymbol{\theta}_t$ if σ_t exceeds the size of the training component.

3. Experiment Results

Recall that the TD method trains a *local* SVM (lSVM) for each leaf of a binary tree, while conventional SVM training constructs *global* SVM (gSVM) classifiers using all the samples for training.

In our experiments, we divided the data sets into two groups. The first group was used to evaluate the efficiency of TD and two alternative methods in terms of speeding up SVM training. The second group was used to verify that the TD method could handle large data sets, for which SVM would take very long time to complete the training process. The first group consisted of six data sets, whose sizes ranged from 10K to 494K, as shown in the first six rows of Table 1. The second group comprised two data sets of size 581K and 4,898K, respectively, as shown in the last two rows of Table 1. The table contains other information about the data sets, including the number of labels, the number of samples, and the features in each data set. All data sets were obtained from the UCI repository. Note that the original "Poker" data set in the repository contains 1 million samples. However, in our experiments, we only used its training subset, whose size is suitable for comparing TD with other methods, including gSVM.

We randomly divided each data set into 6 parts of equal size. Then, we used 4 parts as the training data component, one part as the validation component, and the remaining part as the test component. The TD classifiers were trained on the training and validation components, as described in Section 2.2. On the completion of the training process, we applied the output TD classifier to the corresponding test data set to obtain the test accuracy rate.

Data set	No. of Labels	No. of Samples	No. of Features
Pen Hand Written (PHW)	10	10,992	16
Letter	26	20,000	16
Shuttle	7	58,000	9
Census Income	2	45,222	14
Poker	10	20,843	10
KDD CUP 10%	5	494,021	41
Forest	7	581,012	54
KDD CUP 1999	5	4,898,431	41

Table 1. The data sets used in the experiment.

In each data set, all the feature values of the data points were passed through a normalization procedure. We normalized all the feature values to a real number between 0 and 1. We did this by transforming each value v of feature f into $(v-f_{min})/(f_{max}-f_{min})$, where f_{max} and f_{min} are the maximum and minimum values of f respectively.

We only studied non-linear SVMs in our experiments. Moreover, we used the RBF kernel function to measure the similarity between vectors. As a result, we had two SVM parameters: the penalty factor C, whose values were taken as $\Phi = \{10^a: a = -1, 0, \ldots, 5\}$; and the γ parameter in the RBF function, whose values were taken as $\Psi = \{10^b: b = -4, -3, \ldots, 4\}$. Thus, the set of all SVM parameter values was $\Theta = \Phi \times \Psi$. In all SVM training sessions, we used the LIBSVM software (Hsu and Lin, 2002). We adopted all default options of this software, except the parameter values, which we specified above.

SVM training was implemented under the 1A1 and 1AO approaches. When the 1A1 approach is used, there are $n(n-1)/2$ classifiers, where n is the number of labels. Each of the classifiers assigns one of two possible labels to a given validation/test sample. We used all the classifiers to classify $\mathbf{x}$, a given validation/test sample, based on majority votes. Note that a more efficient technique (Platt et al. 2000) that only requires n classifiers can be used in the validation/testing procedure. However, we adopted the technique developed by Knerr et al. (1990), which requires $n(n-1)/2$ classifiers, because we were only interested in the relative, rather than the absolute, performance of the methods compared in our experiments. When the 1AO approach is used, there are n decision functions, each of which is associated with a label. We assign $\mathbf{x}$ the label associated with the decision function that yields the highest functional value.

To compare TD with existing methods, we implemented two methods designed to speed up SVM training: bagging (an MSR method) and CBD (an SSR method). When implementing bagging, we created a number of SVMs for each $\boldsymbol{\theta} \in \Theta$, where each SVM was trained on 1,500 training samples chosen at random. For each $\boldsymbol{\theta}$, the training was conducted sequentially. We stopped at the first m so that the validation accuracy rate of m SVMs did not exceed that of m-1 SVMs by 0.5%. CBD training comprises two steps: finding a reduced set, and training an SVM on that set for each $\boldsymbol{\theta} \in \Theta$. The first part requires finding k-nearest neighbors of each training sample and deriving the reduced data set via a down-sampling technique. Following Panda et al. (2006), we set k to 100. In finding the 100 nearest neighbors of each training sample $\mathbf{x}$, we keep the current list of 100 nearest neighbors of $\mathbf{x}$. For another training sample $\mathbf{z}$, let $d(\mathbf{x}, \mathbf{z})$ be the distance between $\mathbf{x}$ and $\mathbf{z}$. We need to compare this distance with $d(\mathbf{x}, \mathbf{w})$, where $\mathbf{w}$ is in the current list and holds the largest distance with **x**. Since the squared distance is the sum of squared feature differences, we can speed up the comparison by computing the partial sum of $d^2(\mathbf{x}, \mathbf{z})$. When this partial sum exceeds $d^2(\mathbf{x}, \mathbf{w})$, we stop the comparison and exclude **z** from the current list of **x**. This numerical trick saves a tremendous amount of time in finding the 100 nearest neighbors for all training samples, especially when the number of training samples is large.

The experiment results obtained by TD, bagging, CBD, and gSVM for the first six data sets are shown in Tables 2-6 for 1A1 training, and in Tables 7-11 for 1AO training. The boldface numbers in the tables signify the best performances. Table 2 and Table 7 show the training times of the four methods. The training time of each method comprises the time required to obtain reduced data sets if it is a speedup method, the time to train all SVMs, and the time to search for optimal parameters. The time to input or output data is *not* included, however. The computation for the first seven data sets in Table 1 was conducted on Intel Xeon CPU 3.2 GHz with 2GB RAM. That for "KDD CUP 1999" was conducted on Intel Quad-Core Xeon CPU 2.5 GHz with 8 GB RAM.

Table 3 and Table 8 show *speedup factors* of TD, bagging, and CBD, where the speedup factor of a method $\mathcal{M}$ is computed as gSVM's training time divided by $\mathcal{M}$'s training time. Table 4 and Table 9 show the test accuracy rates of the four compared methods. Note that the TD test accuracy rate is that of the TD classifier with the ceiling size σ_{opt} and SVM-parameters $\boldsymbol{\theta}_{opt}$. When classifying a test sample with SVMs, the most time-consuming part is computing a decision function, whose complexity can be measured in terms of how many SVs are involved in the classification. Therefore, we use the "*number of SVs*" (NSV) as a measure of the complexity, which is the average number of SVs contained in the decision function used to classify a test sample. More specifically, NSV represents the average number of SVs involved in classifying a test sample in the testing process. Note that when a TD classifier is used, the SVs are associated with the leaf that the test sample flows to. Table 5 and Table 10 list the NSVs of the four methods; while Table 6 and Table 11 show the *NSV ratios* of TD, bagging, and CBD, where the NSV ratio of a method $\mathcal{M}$ is computed as gSVM's NSV divided by $\mathcal{M}$'s NSV.

Data set	TD	Bagging	CBD	gSVM
PHW	**275**	2,204	697	1,192
Letter	**768**	7,575	2,598	4,157
Shuttle	**23**	2,100	303	5,096
Census Income	**5,209**	6,100	215,219	315,130
Poker	**5,600**	13,332	992,533	1,307,667
KDD CUP 10%	**371**	17,123	57,789	1,369,600

Table 2. Training times of the four methods, expressed in *seconds*. Training type = 1A1.

Data set	TD	Bagging	CBD
PHW	**4.33**	0.54	1.71
Letter	**5.41**	0.55	1.60
Shuttle	**221.57**	2.43	16.82
Census Income	**60.50**	51.66	1.46
Poker	**233.51**	98.08	1.32
KDD CUP 10%	**3,691.64**	79.99	23.70

Table 3. Speedup factors of TD, bagging, and CBD. Training type = 1A1.

Data set	TD	Bagging	CBD	gSVM
PHW	99.42	99.52	**99.63**	**99.63**
Letter	**97.60**	93.09	95.25	97.54
Shuttle	**99.93**	99.66	99.85	99.92
Census Income	**84.81**	83.75	84.08	84.25
Poker	57.85	56.37	57.50	**58.29**
KDD CUP 10%	**99.96**	99.68	99.91	99.95

Table 4. Test accuracy rates of the four methods. Training type = 1A1.

Data set	TD	Bagging	CBD	gSVM
PHW	**1,140**	38,376	7,785	8,073
Letter	183,450	663,315	**109,376**	183,450
Shuttle	**0.5**	2,465	1,098	1,134
Census Income	**300**	5,381	9,560	10,451
Poker	**5,284**	100,018	145,522	174,942
KDD CUP 10%	**3.9**	1,873	2,783	2,287

Table 5. NSVs of the four methods. Training type = 1A1.

Data set	TD	Bagging	CBD
PHW	**7.08**	0.21	1.04
Letter	1.00	0.28	**1.68**
Shuttle	**2,432.70**	0.46	1.03
Census Income	**34.81**	1.94	1.09
Poker	**33.11**	1.75	1.20
KDD CUP 10%	**584.86**	1.22	0.82

Table 6. NSV ratios of TD, bagging, and CBD. Training type = 1A1.

Data set	TD	Bagging	CBD	gSVM
PHW	**672**	12,271	5,893	19,988
Letter	**3,281**	105,121	87,179	151,476
Shuttle	**36**	5,359	700	36,689
Census Income	**5,191**	6,100	215,219	315,130
Poker	**10,061**	91,058	2,222,084	2,923,776
KDD CUP 10%	**435**	46,879	57,706	2,512,134

Table 7. Training times of the four methods, expressed in *seconds*. Training type = 1AO.

Data set	TD	Bagging	CBD
PHW	**29.74**	1.63	3.39
Letter	**46.17**	1.44	1.74
Shuttle	**1,019.14**	6.85	52.41
Census Income	**60.71**	51.66	1.46
Poker	**290.60**	32.11	1.32
KDD CUP 10%	**5,775.02**	53.59	43.53

Table 8. Speedup factors of TD, bagging, and CBD. Training type = 1AO.

Data set	TD	Bagging	CBD	gSVM
PHW	99.52	99.47	**99.63**	**99.63**
Letter	**97.66**	93.80	96.20	**97.66**
Shuttle	99.89	99.67	99.82	**99.91**
Census Income	**84.81**	83.75	84.08	84.25
Poker	57.62	57.30	56.82	**58.02**
KDD CUP 10%	**99.96**	99.68	99.91	**99.96**

Table 9. Test accuracy rates of the four methods. Training type = 1AO.

Data set	TD	Bagging	CBD	gSVM
PHW	**248**	6,771	1,164	1,210
Letter	16,200	240,394	**14,690**	16,201
Shuttle	**0.4**	1,070	611	266
Census Income	**300**	5,381	9,560	10,451
Poker	**1,801**	160,779	40,965	47,951
KDD CUP 10%	**1.8**	756	1,576	1,566

Table 10. NSVs of the four methods. Training type = 1AO.

Data set	TD	Bagging	CBD
PHW	**4.87**	0.18	1.04
Letter	1.00	0.07	**1.10**
Shuttle	**597**	0.25	0.44
Census Income	**34.81**	1.94	1.09
Poker	**26.63**	0.30	1.17
KDD CUP 10%	**873.37**	2.07	0.99

Table 11. NSV ratios of TD, bagging, and CBD. Training type = 1AO.

We summarize the results shown in Tables 2-11 as follows.

1. In terms of training time, TD outperformed the other three methods on all the data sets (Table 2 and Table 7). Furthermore, TD achieved very large speedup factors for “Shuttle”, “Poker”, and “KDD CUP 10%”, as compared with those derived by bagging and CBD. (Table 3 and Table 8).
2. TD achieved comparable test accuracy rates to those of gSVM on all the data sets. Bagging and CBD lag behind on “Letter” (Table 4 and Table 9).
3. In terms of NSV, TD’s performance was comparable to that of CBD and gSVM on “Letter”, and it outperformed all the other methods on the remaining data sets (Table 5 and Table 10). Moreover, TD obtained very large NSV ratios compared to those of bagging and CBD on “Shuttle”, “Poker”, and “KDD CUP 10%” (Table 6 and Table 11).
4. TD achieved larger speedup factors for 1AO training than for 1A1 training (Table 3 and Table 8), and it yielded lower NSV ratios for 1AO training (Table 6 and Table 11).

To gain insight into how TD achieved its effectiveness, in Table 12, we show the optimal ceiling sizes derived by TD as well as the proportion of training samples that flow to homogeneous leaves. Note that a single table suffices to show all the results, because 1A1 training and 1AO training employ the same decision trees and TD happens to yield the same σ_{opt} value for both approaches.

	PHW	Letter	Shuttle	Census Income	Poker	KDD CUP 10%
σ_{opt}	1,500	24,000	1,500	1,500	1,500	1,500
Proportion	0%	0%	98.42%	5.55%	0%	42.05%

Table 12. The optimal ceiling sizes σ_{opt} obtained by TD and the proportion of training samples that flow to homogeneous leaves.

First, we observe that TD required a low ceiling size, 1,500, on all data sets except “Letter”. This explains why TD generally achieved good speedup factors and NSV ratios. Interestingly, the proportion of training samples that flowed to homogeneous leaves under TD was very high in “Shuttle” and “KDD CUP 10%”. Since no SVM is involved in any homogenous leaves, TD achieved very high speedup factors and NSV ratios on these two data sets. The same fact also explains why TD achieved such low NSVs that even fell below 1 on “Shuttle”. Note that this effect is achieved by decision trees that group neighboring samples into the same leaf. Random decomposition, on the other hand, does not produce the same effect, because the probability that all samples will carry the same label in the same randomly decomposed region is extremely small.

Next, we consider the TD results for “Letter”. In this data set, the optimal ceiling size exceeded the size of the training component. Thus, the output TD classifier was trained on the full data set. However, TD still achieved positive speedup factors. This is because TD trained lSVMs for all the parameter values only on leaves with ceiling size 1,500, which took much less amount of time than training them on the full training component. The amount of time spent on higher ceiling sizes was even smaller, because TD only trained a small number of lSVMs. Moreover, the lSVMs were trained with top-ranked parameters, which tended to require less time than those trained with bottom-ranked parameters.

Table 13 shows the training times required for different ceiling sizes, indicating that TD spends most of its time on the leaves of the lowest ceiling size. In addition, Table 14 shows the test accuracy rates corresponding to different ceiling sizes, assuming that the training was terminated at those sizes. The results demonstrate the benefit of searching for optimal ceiling sizes because, if we terminated the training at ceiling size 1,500 or 6,000, we would obtain significantly lower test accuracy rates.

Data Set	Training Mode	1,500	6,000	24,000
Letter	1A1	633	45	90
	1AO	2,730	178	373

Table 13. The TD training times required for different ceiling sizes.

Data Set	Training Mode	1,500	6,000	24,000
Letter	1A1	95.35	96.61	97.60
	1AO	95.71	96.91	97.66

Table 14. The TD test accuracy rates that correspond to different ceiling sizes.

Data Set	Training Mode	Item	TD
Forest	1A1	Training Time (Sec.)	16,927
		NSV	350
		Test Accuracy Rate (%)	94.61
	1AO	Training Time (Sec.)	26,108
		Number of SVs per Test Sample	289
		Test Accuracy Rate (%)	94.59
KDD CUP 1999	1A1	Training Time (Sec.)	18,834
		NSV	4.7
		Test Accuracy Rate (%)	99.99
	1AO	Training Time (Sec.)	18,550
		NSV	10.4
		Test Accuracy Rate (%)	99.99

Table 15. Training and testing results for the two large data sets.

Finally, Table 15 details the experiment results for the two large data sets. We only conducted TD training on these data sets because training bagging, CBD, or gSVM would require too much time. The TD training for "Forest" took 4.7 and 7.3 hours to complete 1A1 training and 1AO training respectively; while the "KDD CUP 1999" training only took 5.2 hours for both types of training. Further details are given in Table 15.

4. Generalization Error Bounds for the TD Classifier

Let $\mathbf{R}^d$ be the d-dimensional Euclidean space. We assume that a set of training samples $X_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ is given, where $(\mathbf{x}_k, y_k) \in \mathbf{R}^d \times \{-1,1\}$ for $k = 1, \ldots, n$. The

TD method produces a classifier $h(\mathbf{x}, \pi, \mathbf{f})$, where π is a binary tree comprising L leaves, $\mathbf{f} = (f_1, \ldots, f_L)$, and f_i is related to the lSVM trained on leaf i of π, for $i = 1, \ldots, L$. The binary tree π produces a partition function that maps an input in $\mathbf{R}^d$ to $\{1, \ldots, L\}$, and $\pi(\mathbf{x})$ is the leaf that $\mathbf{x}$ flows to. On the other hand, the lSVM trained on leave i, $i = 1, \ldots, L$ is expressed as $f_i \circ \Phi$, where Φ maps an input in $\mathbf{R}^d$ to a Hilbert space $\mathbf{H}$, and f_i is a *linear* function from $\mathbf{H}$ to $\mathbf{R}$. Note that a linear function g can be expressed as

$$g(z) = \langle w, z \rangle$$

for some $w \in \mathbf{H}$ and for all $z \in \mathbf{H}$. For such a function, we define $\| g \| = \langle w, w \rangle^{1/2}$.

If $i = \pi(\mathbf{x})$, then $h(\mathbf{x}, \pi, \mathbf{f}) = sign(f_i(\Phi(\mathbf{x}))$. Let

$$\mathbf{f}^{\pi}(\mathbf{x}) = f_{\pi(\mathbf{x})}(\Phi(\mathbf{x})).$$

It follows that $h(\mathbf{x}, \pi, \mathbf{f}) = sign(\mathbf{f}^{\pi}(\mathbf{x}))$.

Sometimes, Φ is only defined implicitly. That is, instead of specifying the functional form of Φ, only the inner product of $\Phi(\mathbf{u})$ and $\Phi(\mathbf{v})$ is specified as

$$\langle \Phi(\mathbf{u}), \Phi(\mathbf{v}) \rangle = k(\mathbf{u}, \mathbf{v}),$$

where $\mathbf{u}$ and $\mathbf{v} \in \mathbf{R}^d$ and $k(\cdot,\cdot)$ is a kernel function. In the remainder of this section, we assume that the function Φ is given and fixed.

Next, we define several notations. $\mathbf{N}$ is the set of natural numbers; $\mathbf{R}^+$ is the set of positive real numbers; $\mathscr{P}_L(\mathbf{R}^d)$ is the class of all functions from $\mathbf{R}^d$ to $\{1, \ldots, L\}$; $\mathscr{R}(\mathbf{H})$ is the class of all functions from $\mathbf{H}$ to $\mathbf{R}$; and $\mathscr{L}(\mathbf{H})$ is the class of all *linear* functions from $\mathbf{H}$ to $\mathbf{R}$. Moreover, if T is a set, we define $T^L = \{(t_1, \ldots, t_L): t_i \in T \text{ for } i = 1, \ldots, L\}$; that is, T^L comprises all the L-tuples of T's elements.

Using the standard definitions given below, we provide a bound for the generalization error of $h(\mathbf{x}, \pi, \mathbf{f})$ in terms of the shatter coefficient of π and the margin of $\mathbf{f}^{\pi}$. More details can be found in Vapnik (1995), and Cristianini and Shawe-Taylor (2000).

Definition 1. *Let $\mathcal{G} \subseteq \mathscr{P}_L(\mathbf{R}^d)$. For any $n \in \mathbf{N}$, the n^{th} shatter coefficient of $\mathcal{G}$ is*

$$V(\mathcal{G}, n) = \max_{S \subseteq R^d, |S| = n} |\{\pi_S : \pi \in \mathcal{G}\}|,$$

where π_S is the function obtained by restricting π to the domain S.

Definition 2. *Let* $\mathbf{f} = (f_1, \ldots, f_L) \in (\mathcal{R}(\mathbf{H}))^L$, $\pi \in \mathcal{P}_L(\mathbf{R}^d)$, $S \subseteq \mathbf{R}^d \times \{-1,1\}$, *and* $\boldsymbol{\gamma} = (\gamma_1,\ldots, \gamma_L) \in (\mathbf{R}^+)^L$. *We say that* $\mathbf{f}^\pi$ *has margin* $\boldsymbol{\gamma}$ *on S, or* $mg(\mathbf{f}^\pi, S) \geqq \boldsymbol{\gamma}$, *if*

$$y \cdot \mathbf{f}^{\pi}(\mathbf{x}) \equiv y \cdot f_i(\Phi(\mathbf{x})) \geq \gamma_i$$

for any $i \in \{1,\ldots,L\}$ *and any* $(\mathbf{x}, y) \in S$ *with* $\pi(\mathbf{x}) = i$.

We also adopt the following notion of a covering number proposed by Alon et al. (1997).

Definition 3. *Let* $\eta \in \mathbf{R}^+$ *and* $\mathcal{F} \subseteq \mathcal{R}(\mathbf{H})$. *For a subset* $D \subseteq \mathbf{H}$, *let* $\mathcal{C}(\mathcal{F}, D, \eta)$ *be the smallest collection of functions from D to* $\mathbf{R}$ *such that, for each* $f \in \mathcal{F}$, *we have* $g \in \mathcal{C}(\mathcal{F}, D, \eta)$ *with* $| f(z)-g(z) | \leqq \eta$ *for each* $z \in D$. *For* $E \subseteq \mathbf{H}$ *and* $n \in \mathbf{N}$, *we define the covering number of* $\mathcal{F}$ *with respect to E, n, and* η *as*

$$N(\mathcal{F}, E, n, \eta) = \max_{D \subseteq E,\ |D|=n} |\mathcal{C}(\mathcal{F}, D, \eta)|.$$

4.1 Hard Margin Bounds

Given that $\pi \in \mathcal{P}_L(\mathbf{R}^d)$, $\mathbf{f} \in \mathcal{R}(\mathbf{H}))^L$, and the samples in X_n are drawn at random based on the distribution $\mathcal{D}$, the *generalization error* of the classifier $sign(\mathbf{f}^\pi)$ is defined as the probability that $sign(\mathbf{f}^\pi(\mathbf{x})) \neq y$, where $(\mathbf{x}, y)$ is sampled according to $\mathcal{D}$. The following lemma generalizes a known result for SVMs. The proof of the lemma is rather lengthy, so we provide it in Appendix A.

Lemma 1. *Let* $\mathcal{G} \subseteq \mathcal{P}_L(\mathbf{R}^d)$, $\pi \in \mathcal{G}$, $\boldsymbol{\gamma} = (\gamma_1,\ldots, \gamma_L) \in (\mathbf{R}^+)^L$, and $\mathbf{f} = (f_1, \ldots, f_L) \in \mathcal{F}_1 \times \ldots \times \mathcal{F}_L$, *where* $\mathcal{F}_i \subseteq \mathcal{R}(\mathbf{H})$ *and* $1 \leqq i \leqq L$. *For any probability distribution* $\mathcal{D}$ *on* $\mathbf{R}^d \times \{-1, 1\}$, *if the samples in* X_n *are drawn at random based on* $\mathcal{D}$, *then, with probability 1-*δ, *the generalization error of* $sign(\mathbf{f}^\pi)$ *with* $mg(\mathbf{f}^\pi, X_n) \geqq \boldsymbol{\gamma}$ *will be at most*

$$\frac{2}{n}\left[\sum_{i=1}^{L} \log N(\mathcal{F}_i, E, 2n, \gamma_i / 2) + \log V(\mathcal{G}, 2n) + \log(2/\delta)\right],$$

where $E = \{\Phi(\mathbf{x}) : (\mathbf{x}, y) \in \text{supp}(\mathcal{D})\}$ and supp($\mathcal{D}$) *is the support of* $\mathcal{D}$.

Lemma 1 states a general result for $\mathcal{G}$ and $\mathcal{F}_i$ when $i = 1, \ldots, L$. We now consider some special examples of $\mathcal{G}$ and $\mathcal{F}_i$. For $\beta \in \mathbf{R}^+$, we define $\mathcal{L}(\mathbf{H}, \beta)$ as the class of all linear functions $f \in \mathcal{L}(\mathbf{H})$ with $\| f \| \leqq \beta$. A bound can be obtained for the covering number of $\mathcal{L}(\mathbf{H}, \beta)$ with respect to E, n and η, provided that E is a bounded subset of $\mathbf{H}$ (see, for example, Bartlett and Shawe-Taylor, 1988).

Lemma 2. *Let* ρ, β, *and* $\eta \in \mathbf{R}^+$ *and let* $n \in \mathbf{N}$. *Consider any* $E \subseteq \mathbf{H}$ *with* $\| z \| \leqq \rho$ *for every* $z \in E$. *Then, there is a constant c such that*

$$\log N(\mathcal{L}(\mathbf{H}, \beta), E, n, \eta) \le c \frac{\rho^2 \beta^2}{\eta^2} \log^2 n.$$

We also define $\mathcal{B}_L(\mathbf{R}^d)$ as the class of partition functions associated with binary trees with L leaves. Clearly, $\mathcal{B}_L(\mathbf{R}^d) \subseteq \mathcal{P}_L(\mathbf{R}^d)$. The following lemma provides a bound on the n^{th} shatter coefficient of $\mathcal{B}_L(\mathbf{R}^d)$.

Lemma 3. *Let* d, n, *and* $L \in \boldsymbol{N}$. *Then,*

$$\log V(\mathcal{B}_L(\mathbf{R}^d), n) \le L \log(dnL^2).$$

Proof. Consider any n-element subset $S \subseteq \mathbf{R}^d$. The goal is to cut S into L parts. Initially, there is only one part, which is S. We perform the cut operation recursively. Each time, we choose a part of S and cut it into two. To do so, we pick one of d dimensions and one of at most n-1 cutting hyperplanes on that dimension. Thus, there are at most $d(n\text{-}1)(L\text{-}1) \le dnL$ ways to perform one cut operation. To obtain L parts, we repeat the cut operation L-1 times; hence, the number of possible partitions is at most $(dnL)^{L-1} \le (dnL)^L$. Finally, there are $L!$ ways to order the L parts of each partition, yielding the following result:

$$V(\mathcal{B}_L(\mathbf{R}^d), n) \le (dnL)^L \cdot (L!) \le (dnL^2)^L. \qquad \blacksquare$$

From the above three lemmas, we immediately derive the following theorem.

Theorem 1. *Let* ρ *and* $\beta_i \in \mathbf{R}^+$, $1 \le i \le L$, $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_L) \in (\mathbf{R}^+)^L$, $\mathbf{f} = (f_1, \ldots, f_L) \in \mathcal{L}(\mathbf{H}, \beta_1) \times \ldots \times \mathcal{L}(\mathbf{H}, \beta_L)$, *and* $\pi \in \mathcal{B}_L(\mathbf{R}^d)$. *For any probability distribution* $\mathcal{D}$ *on* $\mathbf{R}^d \times \{-1, 1\}$, *if* *the samples in X_n are drawn at random based on $\mathscr{D}$ such that* $\|\Phi(\mathbf{x})\| \leqq \rho$ *for* $(\mathbf{x}, y) \in \mathrm{supp}(\mathscr{D})$*, then, with probability 1-δ, the generalization error of sign*$(\mathbf{f}^{\pi})$ *with mg*$(\mathbf{f}^{\pi}, S) \geqq \boldsymbol{\gamma}$ *will be at most*

$$\frac{c}{n}\left(\sum_{i=1}^{L}\frac{\rho^2\beta_i^2}{\gamma_i^2}\log^2 n + L\log(dnL^2) + \log(1/\delta)\right),$$

for some constant c.

4.2 Soft Margin Bounds

Note that Theorem 1 works in the case where the training data X_n can be separated with a margin vector $\boldsymbol{\gamma}$. If the data is non-separable or noisy, we need to consider the notion of a soft margin (Cristianini and Shawe-Taylor 2000).

Definition 4. *Let* $\mathbf{f} = (f_1, \ldots, f_L) \in (\mathscr{L}(\mathbf{H}))^L$, $\pi \in \mathscr{P}_L(\mathbf{R}^d)$, $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_L) \in (\mathbf{R}^+)^L$, *and* $S \subseteq \mathbf{R}^d \times \{-1, 1\}$. *For* $1 \leqq i \leqq L$, *let* $\{(\mathbf{x}, y) \in X_n: (\mathbf{x}_{i,1}, y_{i,1}), \ldots, (\mathbf{x}_{i,n_i}, y_{i,n_i})\}$ *for some* $n_i \in \mathbf{N}$. *For* $1 \leqq i \leqq L$ *and* $1 \leqq j \leqq n_i$, *let*

$$\xi_{i,j} = \max(0, \gamma_i - y_{i,j} \cdot f_i(\Phi(\mathbf{x}_{i,j}))).$$

The vector $\boldsymbol{\xi}_i = (\xi_{i,1}, \ldots, \xi_{i,n_i})$ *is called the margin slack vector of* f_i *with respect to π and* γ_i *over* X_n *for* $1 \leqq i \leqq L$.

To find the bound for the generalization error in the case of a soft margin, we follow Shawe-Taylor and Cristianini (1999 and 2002). Let $\beta_i \in \mathbf{R}^+$ for $1 \leqq i \leqq L$, $\mathbf{f} = (f_1, \ldots, f_L) \in \mathscr{L}(\mathbf{H}, \beta_1) \times \ldots \times \mathscr{L}(\mathbf{H}, \beta_L)$, $\pi \in \mathscr{P}_L(\mathbf{R}^d)$, $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_L) \in (\mathbf{R}^+)^L$; and let $\boldsymbol{\xi}_i = (\xi_{i,1}, \ldots, \xi_{i,n_i})$ be the slack margin vector of f_i with respect to π and γ_i over X_n for $1 \leqq i \leqq L$. In addition, we assume that $\|\Phi(\mathbf{x})\| \leqq \rho$ for $(\mathbf{x}, y) \in \mathrm{supp}(\mathscr{D})$. We want to construct a space $\hat{\mathbf{H}}$ that has a higher dimension than $\mathbf{H}$ and also form linear functions on $\hat{\mathbf{H}}$ that have desirable margins. For this purpose, we introduce the inner product space

$$\mathscr{I}(\mathbf{H}) = \{f\colon f\colon \mathbf{H} \to \mathbf{R} \text{ and } f \text{ is non-zero on a finite number of inputs}\},$$

where $\langle f, g \rangle = \sum_z f(z)g(z)$ for f and $g \in \mathcal{J}(\mathbf{H})$. Let

$$\hat{\mathbf{H}} = \mathbf{H} \times \mathcal{J}(\mathbf{H}).$$

Since $\mathcal{J}(\mathbf{H})$ is an inner product space, each of its elements defines a linear function on $\mathcal{J}(\mathbf{H})$. Hence, for $f \in \mathcal{L}(\mathbf{H})$ and $g, h \in \mathcal{J}(\mathbf{H})$, the function $(f, g)\mathbf{:}\ \hat{\mathbf{H}} \to \mathbf{R}$, defined by

$$(f, g)(z, h) = f(z) + \langle g, h \rangle,$$

is a linear function. Now, for $1 \leqq i \leqq L$. we define $g_i \in \mathcal{J}(\mathbf{H})$ by

$$g_i = \sum_{j=1}^{n_i} \xi_{i,j} \cdot y_{i,j} \cdot \delta_{\Phi(\mathbf{x}_{i,j})},$$

where

$$\delta_z(z') = \begin{cases} 1 & \text{if } z = z', \\ 0 & \text{otherwise.} \end{cases}$$

We also define $\hat{f}_i : \hat{\mathbf{H}} \to \mathbf{R}$ by

$$\hat{f}_i = (f_i, g_i / \rho).$$

Since $f_i \in \mathcal{L}(\mathbf{H}, \beta_i)$, there exists $w_i \in \mathbf{H}$ with $\| w_i \| \leqq \beta_i$ such that $f_i(z) = \langle w_i, z \rangle$. It follows that for $\hat{z} \in \hat{\mathbf{H}}$,

$$\hat{f}_i(\hat{z}) = \langle \hat{w}_i, z \rangle,$$

where $\hat{w}_i = (w_i, g_i/\rho)$, and

$$\| \hat{w}_i \|^2 \le \| w_i \|^2 + \| \xi_i \|^2 / \rho^2 \le \beta_i^2 + \| \xi_i \|^2 / \rho^2.$$

Therefore, $\hat{f}_i \in \mathcal{L}(\hat{\mathbf{H}}, \hat{\beta}_i)$, where $\hat{\beta}_i = \sqrt{\beta_i^2 + \| \xi_i \|^2 / \rho^2}$. We now define $\tau_\rho : \mathbf{H} \to \hat{\mathbf{H}}$ by

$$\tau_\rho(z) = (z, \rho \cdot \delta_z).$$

Then, for $(\mathbf{x}_{i,j}, y_{i,j}) \in X_n$,

$$\begin{aligned} y_{i,j} \cdot \hat{f}_i(\tau_\rho(\Phi(\mathbf{x}_{i,j}))) &= y_{i,j} \cdot f_i(\Phi(\mathbf{x}_{i,j})) + y_{i,j} \cdot (\xi_{i,j} \cdot y_{i,j}) \\ &= y_{i,j} \cdot f_i(\Phi(\mathbf{x}_{i,j})) + \xi_{i,j} \\ &\geq \gamma_i. \end{aligned}$$

Let

$$\Psi = \tau_\rho \circ \Phi,$$

$$\hat{\mathbf{f}} = (\hat{f}_1, \ldots, \hat{f}_L), \text{ and}$$

$$\hat{\mathbf{f}}^\pi(\mathbf{x}) = \hat{f}_i(\Psi(\mathbf{x})),$$

where $i = \pi(\mathbf{x})$. From the above discussion, $\hat{\mathbf{f}}$ has a margin $(\gamma_1, \ldots, \gamma_L)$ with respect to π on X_n. Also, for $(\mathbf{x}, y) \in \text{supp}(\mathcal{D})$,

$$\|\Psi(\mathbf{x})\|^2 \leq \|\Phi(\mathbf{x})\|^2 + \|\rho\delta_{\Phi(\mathbf{x})}\|^2 \leq 2\rho^2.$$

Finally, we observe that, for any $(\mathbf{x}, y) \notin X_n$, $sign(\mathbf{f}^\pi) = sign(\hat{\mathbf{f}}^\pi(\mathbf{x}))$; therefore, $sign(\mathbf{f}^\pi)$ and $sign(\hat{\mathbf{f}}^\pi)$ have the same generalization error on inputs that do not fall within X_n. However, it is possible that $sign(\mathbf{f}^\pi)$ will make more mistakes on X_n, and thus has a larger generalization error over $\mathcal{D}$ than $sign(\hat{\mathbf{f}}^\pi)$. As suggested by Shaw-Taylor and Cristianini (2002), this can be handled by modifying $sign(\mathbf{f}^\pi)$ on the misclassified elements in X_n. We call this new function the X_n-filtered version of $sign(\mathbf{f}^\pi)$.

Now we can apply Theorem 1, with the space $\mathbf{H}$ replaced by $\hat{\mathbf{H}}$ and the mapping Φ replaced by Ψ, to obtain a bound on the generalization error of $sign(\hat{\mathbf{f}}^\pi)$, but with the quantity $\rho^2\beta^2/\gamma^2$ replaced by

$$\frac{2\rho^2(\beta_i^2 + \|\xi_i\|^2/\rho^2)}{\gamma_i^2} = \frac{2(\rho^2\beta_i^2 + \|\xi_i\|^2)}{\gamma_i^2}.$$

Since β_i and γ_i are interdependent quantities, $1 \leq i \leq L$, we can fix one of them and seek to optimize the other. In the formulation of the SVM optimization problem, the objective is set to minimize β_i under the constraint that $\gamma_i = 1$ for $1 \le i \le L$. Under this convention, we obtain the following theorem.

Theorem 2. *Let* $\rho \in \mathbf{R}^+$, $\mathbf{f} \in \mathcal{L}(\mathbf{H}_1, \beta_1)\times\ldots\times\mathcal{L}(\mathbf{H}_L, \beta_L)$, *and* $\pi \in \mathcal{B}_L(\mathbf{R}^d)$. *Consider any probability distribution* $\mathcal{D}$ *on* $\mathbf{R}^d\times\{-1, 1\}$ *such that* $||\Phi(\mathbf{x})|| \leqq \rho$ *for* $(\mathbf{x}, y) \in \mathrm{supp}(\mathcal{D})$. *If the samples in* X_n *are drawn at random based on* $\mathcal{D}$, *then, with probability* $1\text{-}\delta$, *the generalization error of the* X_n*-filtered version of sign*$(\mathbf{f}^\pi)$ *will be at most*

$$\frac{c}{n}\Big(\sum_{i=1}^{L}\big(\rho^2\beta_i^2 + \|\xi_i\|^2\big)\log^2 n + L\log(dnL^2) + \log(1/\delta)\Big), \tag{2}$$

for some constant c, where for $1 \le i \le L$, ξ_i *is the margin slack vector of* f_i *with respect to* π *and* γ_i *over* X_n.

4.3 A Numerical Investigation

Recall that a TD classifier is associated with a tree with L leaves and each leaf is associated with an lSVM. If the tree has only one leaf (i.e., $L = 1$), then the TD classifier will be reduced to an SVM classifier, which has the following generalization error bound

$$\frac{c}{n}\Big(\big(\rho^2\beta^2 + \|\xi\|^2\big)\log^2 n + \log(1/\delta)\Big) \tag{3}$$

(cf. Cristianini and Shawe-Taylor, 2000).

Let us compare the terms that appear in parentheses in (2) and (3). The second term $L\log(dnL^2)$ in (2) is the shatter coefficient of the partition function π associated with a binary tree. We claim that (2) is dominated by the first term, which is the sum of L quantities associated with L leaves of a binary tree, as opposed to the single quantity in (3). Moreover, the first term in (2) is comparable to the corresponding quantity in (3). The above two claims are confirmed by the following experiment results.

To validate the first claim, we compared the two leading terms that appear in parentheses in (2). The first term is $T_1 = \sum_{i=1}^{L}\big(\rho^2\beta_i^2 + \|\xi_i\|^2\big)\log^2 n$ and the second term is $T_2 = L\log(dnL^2)$. The results, shown in Table 16, confirm the claim that T_1 far exceeds T_2 and (2) is dominated by T_1. Note that we compute T_1 under the following assumptions. (i) When the data set contains more than two labels, T_1 is taken as the average of such quantities over all classifiers. (ii) The value of ρ is always 1 when RBF kernels are involved. (iii) The value of $\| \xi \|^2$ is obtained from the solution to the quadratic programming optimization problem. Further details can be found in Cristianini and Shawe-Taylor (2000), Section 6.1.2.

Data Set	1A1		1AO	
	T_1	T_2	T_1	T_2
PHW	1,699	55	5,769	55
Shuttle	1,152,495	110	2,756,183	110
Census Income	101,177	481	101,177	481
Poker	16,403	194	62,638	194
KDD CUP 10%	1,297,678	287	57,426	287

Table 16. The values of two leading terms T_1 and T_2 that appear in the generalization error bounds for TD classifiers. Training types = 1A1 and 1AO.

To validate the second claim, we compare $R = \sum_{i=1}^{L} \left(\rho^2 \beta_i^2 + \| \xi_i \|^2 \right)$ and $S = \rho^2 \beta^2 + \| \xi \|^2$, which are derived, respectively, from the first terms in the generalization error bounds for TD and gSVM classifiers (i.e., in (2) and (3)) with the common factor $\log^2 n$ removed from them. To make a meaningful comparison between R and S, both classifiers have to take the same (C, γ) values, which we specify as the optimal values for gSVM. As a result, for some data sets, we had to train new TD classifiers, using the same decomposition schemes (i.e., the same binary trees and same ceiling sizes) as the old classifiers, but different (C, γ) values. Table 17 shows the values of S, R, and R/L, derived from two data sets. Note that "Letter" is not included in the table because the TD classifier using the designated values of (C, γ) would be the same as the gSVM for this data set. It is clear that the values of R are as small (less than 1,500) as, or of the same order of magnitude as, those of the corresponding S. In fact, S can be viewed as the slack-to-margin ratio and R as the sum of such ratios. The results show that each lSVM classifier generates smaller slack-to-margin ratios than the corresponding gSVM classifier, while the sum of lSVM ratios is comparable to the corresponding gSVM ratio. This explains why the test accuracy rates of TD classifiers are comparable to those of gSVM classifiers.

Data Set	1A1			1AO		
	S	*R*	*R/L*	*S*	*R*	*R/L*
PHW	74	133	17	326	450	56
Shuttle	114,786	75,630	5,402	593,967	180,867	12,919
Census Income	8,211	6,800	128	8,211	6,800	128
Poker	464	1,143	48	2,899	4,366	182
KDD CUP 10%	100,227	70,796	2,212	5,089	3,133	98

Table 17. The values of *S* and *R*, which appear in the generalization error bound for the gSVM and the TD classifiers respectively, and the values of *R*/*L*. Training types = 1A1 and 1AO.

5. Conclusion

We have proposed a method that uses a binary tree to decompose a data space and trains an lSVM on each of the decomposed regions. The resultant TD classifier can be constructed in a much shorter time than the corresponding gSVM classifier, and still achieve comparable accuracy rates to the latter. We also provide a generalization error bound for the TD classifier. Using some data sets to compute the theoretical bounds for gSVM and TD classifiers, we find that TD classifiers generate comparable error bounds than gSVM classifiers. This finding explains why TD classifiers can achieve more or less the same accuracy rates as gSVM classifiers, even though the training times are small relative to the gSVM training times.

References

N. Alon, S. Ben-David, N. Cesa-Bianchi, and D. Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. In *Journal of ACM*, 44(4): 615–631, 1997.

P. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1998.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. In *Journal of Machine Learning Research*, 6:1579-1619, 2005.

L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of classifier methods: A case study in handwriting digit recognition. In *Proc. Int. Conf. Pattern Recognition*, pages 77–87, 1994.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall, 1984.

L. Breiman. Bagging predictors. In *Machine Learning*, 262:123-140, 1996.

C. J. C. Burges, A tutorial on support vector machines for pattern recognition. In *Data Mining and Knowledge Discovery*, 2(2): 1-47, 1998.

R. Colbert, S. Bengio, and Y. Bengio, A parallel mixture of SVMs for very large scale problems. In Neural Computation, 14(5): 1105-1114, 2002.

C. Cortes and V. Vapnik. Support vector machines. In *Machine Learning*, 20: 1-25, 1995.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

L. Devroye, L Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

T. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. In *Machine Learning*, 40: 139-157, 2000.

T. Glasmachers and C. Igle. Maximum-gain working set selection for SVMs. In *Journal of Machine Learning Research*, 7: 1437–1466, 2006.

H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik. Parallel support vector machines: the cascade SVM. In L. K. Saul, Y. Weiss and L. Bottou, editors, *Advances in Neural Information Processing Systems*. MIT Press, 2004.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. In *IEEE Transactions on Neural Networks*, 13(2): 415-425, 2002.

S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: A stepwise procedure for building and training a neural network. In J. Fogelman, editor, *Neurocomputing: Algorithms, Architectures and Applications*. Springer-Verlag, 1990.

N. D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: the informative vector machine. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*. MIT Press, 2003.

D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI Repository of machine learning databases, Irvine, CA: University of California, Department of Information and Computer Science, 1998. [http://www.ics.uci.edu/~mlearn/ MLRepository.html].

E. Osuna, R. Freud, and F. Girosi. An improved training algorithm for support vector machines. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII*, pages 276-285, 1997.

N. Panda, E. Y. Chang, and G. Wu. Concept boundary detection for speeding up SVMs. In *Proceedings of International Conference on Machine learning*, pages 681-688, 2006.

D. Pavlov, D. Chudova, and P. Smyth. Towards scalable support vector machines using squashing. In *ACM SIGKDD*, pages 295-299, 2000.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. J. Smola, editors, .*Advances in Kernel Methods: Support Vector Learning.* MIT Press, 1999.

J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In S. A. Solla, T. K. Leen and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*. MIT Press, 2000.

J. R. Quinlan. Induction of Decision Tree. In *Machine Learning*, 1(1): 81-106, 1986.

A. Rida, A. Labbi, and C. Pellegrini. Local experts combination through density decomposition. In *Proceedings of International Workshop on AI and Statistics*, 1999.

R. E. Schapire. The strength of weak learnability. In *Machine Learning*, 5: 197-227, 1990.

R. E. Schapire and Y. Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3): 135-168, 2000.

A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of 7th International Conference on Machine Learning*, pages 911-918, 2000.

B. Schölkopf, C. J. C. Burges, and A. J. Smola. *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

J. Shawe-Taylor and N. Cristianini. Margin distribution bounds on generalization. In *Proceedings of the European Conference on Computational Learning Theory*, pages 263-273, 1999.

J. Shawe-Taylor and N. Cristianini. On the generalization of soft margin algorithms. In *IEEE Transactions on Information Theory*, 48(10): 2721–2735, 2002.

A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the International Conference on Machine Learning*, pages 911–918, 2000.

V. Tresp. A Bayesian committee machines. *Neural Computation*, 12, pages 2719-2741, 2000

V. Tresp. Scaling kernel-based systems to large data sets. In *Data Mining and Knowledge Discovery*, 5: 197-211, 2001.

V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Theory of Probability and Its Applications*, 16: 264-280, 1971.

V. Vapnik and A. Chervonenkis. Ordered risk minimization I. In *Automation and Remote Control*, 35: 1226-1235, 1974.

V. Vapnik and A. Chervonenkis. Ordered risk minimization II. In *Automation and Remote Control*, 35: 1403-1412, 1974.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.

H. Yu, J. Yang, J. Han, and X. Li. Making SVMs scalable to large data sets using hierarchical cluster indexing. In *Data Mining and Knowledge Discovery*, 11: 295-321, 2003.

Appendix A. Proof of Lemma 1

Let $\mathcal{G} \subseteq \mathcal{P}_L(\mathbf{R}^d)$, $\boldsymbol{\gamma} = (\gamma_1,\ldots, \gamma_L) \in (\mathbf{R}^+)^L$, and $\mathcal{F}_i \subseteq \mathcal{R}(\mathbf{H})$ for $i = 1, \ldots, L$. The samples in X_n are drawn at random based on the distribution $\mathcal{D}$.

Our goal is to find an upper bound for the probability of the following event:

- A1: *there exist* $\pi \in \mathcal{G}$ *and* $\mathbf{f} = (f_1,\ldots, f_L) \in \mathcal{F}_1\times\ldots\times\mathcal{F}_L$ *such that* $mg(\mathbf{f}^\pi, X_n) \geqq \boldsymbol{\gamma}$ *and* $err(\mathbf{f}^\pi, \mathcal{D})>\varepsilon$, *where* $err(\mathbf{f}^\pi, \mathcal{D})$ *is the probability that* $sign(\mathbf{f}^\pi(\mathbf{x})) \neq y$ *with* $(\mathbf{x}, y)$ *will be sampled according to* $\mathcal{D}$.

We relate event A$_1$ to another event A$_2$ in which an additional set $\hat{X}_n$ of n independent samples are drawn at random based on $\mathcal{D}$:

- A2: *there exist* $\pi \in \mathcal{G}$ *and* $\mathbf{f} = (f_1,\ldots, f_L) \in \mathcal{F}_1\times\ldots\times\mathcal{F}_L$ *such that* $mg(\mathbf{f}^\pi, X_n) \geqq \boldsymbol{\gamma}$ *and* $err(\mathbf{f}^\pi, \hat{X}_n) > \varepsilon/2$, *where* $err(\mathbf{f}^\pi, \hat{\hat{X}}_n) = |\{(\mathbf{x}_i, y) \in X_n : sign(\mathbf{f}^\pi(\mathbf{x})) \neq y\}|/n$.

By a standard argument (cf. Vapnik 1998), one can show that

$$\Pr_{X_n}\left[\mathrm{A}_1\right] \leq 2\cdot \Pr_{X_n,\hat{X}_n}\left[\mathrm{A}_2\right].$$

The next step involves finding a bound for $\Pr_{X_n,\hat{X}_n}[\mathrm{A}_2]$. To do this, we observe that $\Pr_{X_n,\hat{X}_n}[\mathrm{A}_2]$ equals the probability of another event, A$_3$, where a set X_{2n} of $2n$ samples are drawn at random based on $\mathcal{D}$, and X_{2n} is further divided randomly into two disjoint parts of equal size: W_1 and W_2.

- A3: *there exist* $\pi \in \mathcal{G}$ *and* $\mathbf{f} = (f_1,\ldots, f_L) \in \mathcal{F}_1\times\ldots\times\mathcal{F}_L$ *such that* $mg(\mathbf{f}^\pi, W_1) \geqq \boldsymbol{\gamma}$ *and* $err(\mathbf{f}^\pi, W_2) > \varepsilon/2$.

Let $\mathcal{G}(X_{2n})$ be the family of functions of $\mathcal{G}$ restricted to the domain $\{\mathbf{x}: (\mathbf{x}, y) \in X_{2n}\}$; and for $\pi \in \mathcal{G}(X_{2n})$, let

$$B^\pi_{\gamma/2}(\Phi(X_{2n})) = \mathcal{C}(\mathcal{F}_1, \Phi(X^{(1)}_{2n}), \gamma_1/2)\times\ldots\times\mathcal{C}(\mathcal{F}_L, \Phi(X^{(L)}_{2n}), \gamma_L/2),$$

where $\Phi(X_{2n}^{(i)}) = \{\Phi(\mathbf{x}): (\mathbf{x}, y) \in X_{2n}, \pi(\mathbf{x}) = i\}$. For $\mathbf{g} = (g_1, \ldots, g_L) \in B_{\gamma/2}^{\pi}(\Phi(X_{2n}))$, let $g^{\pi}(\mathbf{x}) = g_i(\mathbf{x})$ for $\mathbf{x} \in \Phi(X_{2n}^{(i)})$ for $1 \leqq i \leqq L$. Then, for $\pi \in \mathcal{G}(X_{2n})$ and $\mathbf{f} = (f_1, \ldots, f_n) \in \mathcal{F}_1 \times \ldots \times \mathcal{F}_L$, there exists $\mathbf{g} = (g_1, \ldots, g_L) \in B_{\gamma/2}^{\pi}(\Phi(X_{2n}))$ such that for any $(\mathbf{x}, y) \in X_n$, if $\pi(\mathbf{x}) = i$, then

$$| f_i(\Phi(\mathbf{x})) - g_i(\Phi(\mathbf{x})) | \leq \gamma_i/2.$$

For such $\mathbf{f}^{\pi}$ and $\mathbf{g}^{\pi}$, $mg(\mathbf{f}^{\pi}, W_1) \geqq \boldsymbol{\gamma}$ implies that $mg(\mathbf{g}^{\pi}, W_1) \geqq \boldsymbol{\gamma}/2$; and $err(\mathbf{f}^{\pi}, W_2) \geqq \varepsilon/2$ implies that $err_{\gamma/2}(\mathbf{g}^{\pi}, W_2) \geqq \varepsilon/2$, where $err_{\gamma/2}(\mathbf{g}^{\pi}, W_2)$ is the proportion of $(\mathbf{x}, y)$ in W_2 for which $g_i(\mathbf{x}) < \gamma_i/2$ if $\pi(\mathbf{x}) = i$. Therefore, the probability of the event A$_3$ cannot exceed that of the following event A$_4$.

- A4: *there exist* $\pi \in \mathcal{G}$ *and* $\mathbf{g} \in B_{\gamma/2}^{\pi}(\Phi(X_{2n}))$ *such that* $mg(\boldsymbol{g}^{\pi}, W_1) \geqq \boldsymbol{\gamma}/2$ *and* $\mathrm{err}_{\gamma/2}(\boldsymbol{g}^{\pi}, W_2) \geqq \varepsilon/2$.

For any X_{2n}, π, and g, we have

$$\Pr_{W_1,W_2}\left[mg(\mathbf{g}^{\pi}, W_1) \geq \gamma/2 \text{ and } err_{\gamma/2}(\mathbf{g}^{\pi}, W_2) \geq \varepsilon/2\right] \leq \frac{\binom{2n-(\varepsilon/2)n}{n}}{\binom{2n}{n}} \leq \left(\frac{n}{2n}\right)^{(\varepsilon/2)n} = 2^{-\varepsilon n/2}.$$

Therefore, we have

$$\Pr_{X_{2n},W_1,W_2}\left[\mathrm{A}_4\right] = \mathrm{E}_{X_{2n}}\left[\Pr_{W_1,W_2}\left[\mathrm{A}_4\right]\right] \leq \mathrm{E}_{X_{2n}}\left[\sum_{\pi\in\mathcal{G}(X_{2n})} \sum_{g\in B_{\gamma/2}^{\pi}} 2^{-\varepsilon n/2}\right].$$

Since $|X_{2n}| = 2n$, $|\mathcal{G}(X_{2n})| \leqq V(\mathcal{G}, 2n)$. Moreover,

$$|B_{\gamma/2}^{\pi}(X_{2n})| = \prod_{1\leq i\leq L} |\mathcal{C}(\mathcal{F}_i, X_{2n}^{(i)}, \gamma_i/2)| \leq \prod_{1\leq i\leq L} N(\mathcal{F}_i, E, |X_{2n}^{(i)}|, \gamma_i/2) \leq \prod_{1\leq i\leq L} N(\mathcal{F}_i, E, 2n, \gamma_i/2),$$

where $E = \{\Phi(\mathbf{x}): (\mathbf{x}, y) \in \mathrm{supp}(\mathcal{D})\}$. As a result, we have

$$\Pr_{X_n}\left[\mathrm{A}_1\right] \leq 2 \cdot \Pr_{X_{2n},W_1,W_2}\left[\mathrm{A}_4\right] \leq 2 \cdot V(\mathcal{G}, 2n) \cdot \left(\prod_{1\leq i\leq L} N(\mathcal{F}_i, E, 2n, \gamma_i/2)\right) \cdot 2^{-\varepsilon n/2}.$$

The last quantity is δ, if

$$\varepsilon = \frac{2}{n}\left(\log V(\mathcal{G}, 2n) + \sum_{1 \le i \le L} \log N(\mathcal{F}_i, E, 2n, \gamma_i / 2) + \log(2/\delta) \right).$$

This proves the lemma.