A Hierarchical Method for Multi-Class Support Vector Machines

Volkan Vural VVURAL@ECE.NEU.EDU
Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115 USA

Jennifer G. Dy JDY@ECE.NEU.EDU
Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115 USA

Abstract

We introduce a framework, which we call Divide-by-2 (DB2), for extending support vector machines (SVM) to multi-class problems. DB2 offers an alternative to the standard one-against-one and one-against-rest algorithms. For an N class problem, DB2 produces an $N-1$ node binary decision tree where nodes represent decision boundaries formed by $N-1$ SVM binary classifiers. This tree structure allows us to present a generalization and a time complexity analysis of DB2. Our analysis and related experiments show that, DB2 is faster than one-against-one and one-against-rest algorithms in terms of testing time, significantly faster than one-against-rest in terms of training time, and that the cross-validation accuracy of DB2 is comparable to these two methods.

1. Introduction

The Support Vector Machine (SVM) is a learning approach that implements the principle of Structural Risk Minimization (SRM). Basically, SVM finds a hyper-plane that maximizes the margin between two classes.

SVM was originally designed by Vapnik (1995) for binary classification Yet, many applications have more than two categories. There are two ways for extending SVMs to multi-class problems: (1) consider all the data in one optimization problem. Related research can be found in (Crammer & Singer, 2000; Weston & Watkins, 1999), or (2) construct several binary classifiers. One can formulate the multi-class data into one optimization problem, but since the dominating factor that contributes to the time complexity for training the algorithm is the number of data samples that exist in the optimization problem, algorithms in category (1) are significantly slower than the ones that include several binary classifiers where each classifier classifies a small portion of data. A comparison of the training time for the different methods is given in (Hsu & Lin,).

Currently, there exist two popular algorithms to construct and combine several SVMs for N-class problems. The first one, which is also known as the standard method(Vapnik, 1998), includes N different classifiers where N is the number of classes. The i^{th} classifier is trained while labeling all the samples in the i^{th} class as positive and the rest as negative. We will refer to this algorithm as one-against-rest throughout this paper. The second algorithm, proposed by Knerr et al. (1990), constructs $N \times (N-1)/2$ classifiers, using all the binary pairwise combinations of the N classes. We will refer to this as one-against-one SVMs. To combine these classifiers, while Knerr et al. (1990), suggested using an AND gate, Friedman (1996) suggested Max Wins algorithm that finds the resultant class by first voting the classes according to the results of each classifier and then choosing the class which is voted most. Platt et al. (2000) proposed another algorithm in which Directed Acyclic Graph is used to combine the results of one-against-one classifiers (DAGSVM).

Dumais and Chen (2000) worked on a hierarchical structure of web content in which natural hierarchies exist. They divided the problem into two levels. In the first level they grouped similar classes under some main topics and called these top-level categories. To distinguish the categories from each other, they used one-against-rest algorithm. In the second level, models are learned to distinguish each category from only those categories within the same top-level category again using one-against-rest method. They also applied different feature sets for different levels.

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004. Copyright 2004 by the authors.

In this paper, we introduce a new strategy for extending SVMs to multi-class problems: divide-by-2 (DB2). One of the most important advantages of DB2 is its flexibility. It offers various options in its structure so that one can modify and adapt the algorithm according to the needs of the problem, which makes it preferable against the other existing methods. Another advantage of DB2 is that it creates only $N-1$ binary classifiers. This property of DB2, combined with its tree structure, makes it very fast in terms of testing time compared to the other algorithms. Moreover, the standard one-against-one and one-against-rest algorithms do not have a formulation for an error bound. On the other hand, the tree structure of DB2 let us present an error bound similar to the one derived for DAGSVM.

In section 2, we describe how to train and test with DB2 and present several options that DB2 offers. We analyze the time complexity of our algorithm in section 3 and generalization error in section 4. In section 5, we present an adaptive way that can be applied to every multi-class algorithm. In section 6, we report the experimental results comparing the accuracy and time performances of the algorithms. We provide our conclusions and suggest directions for future research in section 7.

2. Divide-by-2 Method

Starting from the whole data set, DB2 hierarchically divides the data into two subsets until every subset consists of only one class. DB2 divides the data such that instances belonging to the same class are always grouped together in the same subset. Thus, DB2 requires only $N-1$ classifiers. In section 2.1 we describe in detail how these $N-1$ classifiers are built during training. And, in section 2.2 we illustrate how DB2 classifies new data in the testing phase.

2.1. Training

The basic strategy is to divide the data into two subsets at every hierarchical level. How do we group the N classes into two? Different criteria can be used for division. The best way is to group them such that the resulting subsets have the largest margin. This requires C_2^N comparisons and SVM classifications which defeats our purpose of building as few classifiers as possible. Instead, we consider the division step as a clustering problem. One method is to use k-means clustering (Forgy, 1965). An even simpler method is to divide them based on their class mean distances from the origin (Method 2). One may also wish to group the classes according to other criteria, such as speed of implementation (Method 3). One can also

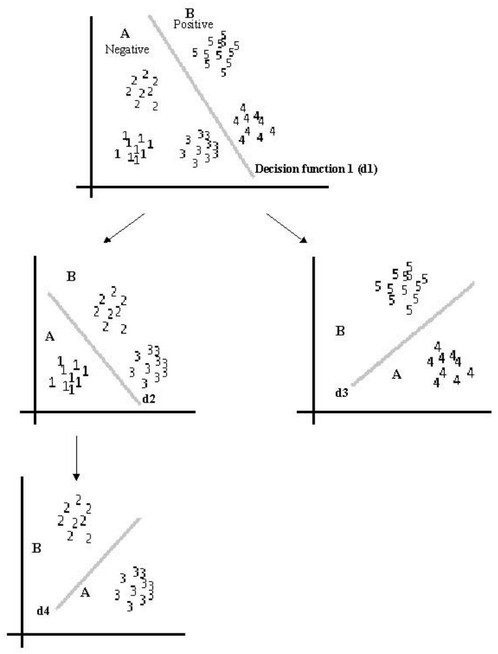

Figure 1. Training

think of other ways of splitting the data.

Method 1: k-means based division

We represent each class with its corresponding mean (μ_i) defined by,

$$\mu_j = \frac{1}{m_j} \sum_{x_i \epsilon \omega} x_i, \tag{1}$$

where m_j is the number of data points in class ω and x_i is a data vector. We, then, group the N $\mu_j's$ into two, using the k-means algorithm.

Method2 : Spherical shells

Let μ_j be the mean of the data belonging to class j, and the total mean, M, as

$$M = \frac{1}{m} \sum_{i=1}^{m} x_i \tag{2}$$

where m is the total number of data points. Using M as a threshold, we group the classes with μ_j smaller than M as the negative class, and the others as the positive class. In three dimensions, separation can be visualized as drawing a sphere separating the space into two parts, and labeling the classes with μ_j inside the sphere as negative and the ones outside as positive.

Method 3: Balanced Subsets

We divide the data into two subsets such that the difference in the number of the samples in each subset is minimum. This criteria is useful if the speed of the process has a high importance or the data has a skewed class distribution.

We summarize the training phase of DB2 as follows:

1. Using one of the methods mentioned above, divide all the data samples into two subsets, A and B.

2. Apply SVM to A and B and find the parameters of the decision boundary separating them.

3. Repeat the steps for both A and B until all the subclasses include only one class.

Figure 1 illustrates the algorithm flow of the training process for a five class data sample.

2.2. Testing

DB2 training leads to a binary decision tree structure for testing. Figure 2 illustrates the decision tree that we built for the testing phase of the five class problem depicted in Figure 1.

At the beginning, all the classes are assumed to be nominees of the true class. At every node, after applying the corresponding decision function to the test input, the nominees that do not exist in the region (positive or negative) in which the test input belongs, are eliminated. Following the branches that indicate the same labels as the result of the decisions, we end up with the predicted class.

The best case occurs if we find the predicted class at the first node, and the worst case occurs if we find the predicted class after applying all the $N-1$ decision functions. In one-against-one, a test data is applied to all $N \times (N-1)/2$ classifiers. For one-against-rest exactly N classifiers and for DAGSVM exactly $N-1$ classifiers are applied. That is why we expect DB2 to be faster than all other algorithms in terms of testing time.

3. Time Complexity

The quadratic optimization problem in the training phase of SVM, slows down the training process. Platt (1998) introduced a fast algorithm, which is called SMO, for training support vector machines. Using SMO, training a single SVM is observed to scale in polynomial time with the training set size m:

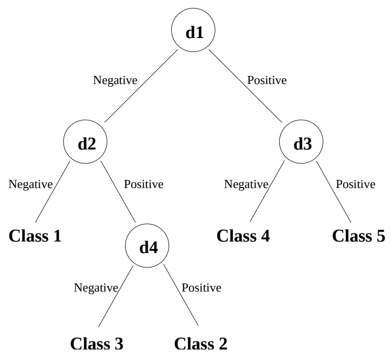

Figure 2. Testing

$$T_{single} = cm^{\gamma} \tag{3}$$

With this relation, we can find the training time for one-against-rest as

$$T_{1-v-rest} = cNm^{\gamma} \tag{4}$$

From equation 3, the training time for one-against-one is found as

$$T_{1-v-1} = T_{DAGSVM} = 2^{\gamma-1}cN^{2-\gamma}m^{\gamma} \tag{5}$$

assuming that the classes have the same number of training data samples. With the same assumption, we can obtain a balanced tree in DB2 using the first method mentioned in section 2.1. Therefore at any i^{th} level of the tree (i=0,1,2,... $\lfloor log_2 N-1 \rfloor + 1$), the training time would be

$$T_{i^{th}level} = 2^{i}c\Big(\frac{m}{2^{i}}\Big)^{\gamma} \tag{6}$$

The total training time becomes

$$T_{DB2} \leq \sum_{i=0}^{\lfloor log_2 N-1 \rfloor + 1} cm^{\gamma}\left(\frac{2}{2^{\gamma}}\right)^{i} \tag{7}$$

which can be proved to be

$$T_{DB2} \quad \leq \quad cm^{\gamma}\frac{2^{\gamma-1}}{2^{\gamma-1}-1}$$

In (Platt et al., 2000), they assumed that the typical value for γ is 2. In this case, one-against-one methods and DB2 have the same time complexity for training.

$$T_{DAGSVM} = T_{1-v-1} = T_{DB2} = 2cm^{\gamma}$$

For balanced data sets, DB2 and one-against-one algorithms are close to each other in terms of time complexity, and they are relatively faster than 1-against-rest. On the other hand, if the training data is unbalanced, DB2 becomes faster than one-against-one methods. For instance, if there is one large class and $N-1$ other small classes, we can separate the large class at the first level of the tree, and the rest of the classifiers will be trained using the small classes only. In a one-against-one approach, the large class will contribute to N classifiers, which will slow down the training process. Related experimental results are provided in section 6.

4. Generalization Analysis

A nice property of the DB2 framework is that an error bound can be obtained, unlike the regular one-against-one and one-against-rest methods except for the DAGSVM implementation of one-against-one. As explained in section 2, DB2 forms a decision tree that is acyclic and directed for testing. A Vapnik Chervonenkis (VC) analysis of directed acyclic graphs is presented and an error bound is provided in Theorem 2 in (Platt et al., 2000), using the results derived in (Bennett et al., 2000).

According to the theorem, if we are able to correctly distinguish class j from the other classes in a random m-sample with a directed decision graph of a decision tree G over N classes containing $N-1$ decision nodes with margins γ_i at node i, then with probability $1-\sigma$,

$$\epsilon_j(G) \leq \frac{130R^2}{m}(D' log(4em) log(4m) + log\frac{2(2m)^T}{\sigma})$$

where $\epsilon_j(G) = P\{$x:x is misclassified as class j by $G\}$, $D' = \sum_{i\in j-nodes}\frac{1}{\gamma_i^2}$, $T \leq N-1$ and R is the radius of a ball containing the support of the distribution.

Observe that the error bound changes depending on γ_i's and T's for DAGSVM and DB2. In DAGSVMs $T = N-1$, which is the worst case for DB2, and the best case for DB2 is only $T = 1$. On the other hand, the margin at each node is an unpredictable variable depending on the kernel function, which makes us unable to compare the error bounds for the two methods.

5. Adaptive Approach

Maximizing the margin between two classes and the usage of kernel functions are two of the main building blocks of SVMs. Kernel functions offer an alternative solution by mapping the data into a higher dimensional feature space in which we can distinguish the data more easily. There are different options for kernel functions depending on the distribution of the training data, but among various kernel functions, how should one choose the best? The generalization ability of a machine can be used as a criterion. To control the generalization ability of a machine, one has to minimize the expectation of the test error, which can be achieved by minimizing the following criterion (Vapnik, 1998):

$$R(D, w) = D^2|w|^2 \qquad (8)$$

where D is the radius of the smallest sphere that includes the training vectors,which is given as:

$$D^2 = \sum_{i,j=1}^{l}\beta_i\beta_j K(x_i, x_j) \qquad (9)$$

and $|w|$ is the norm of the weights of the hyper-plane in feature space, which is obtained as:

$$|w| = \sum_{i,\alpha}^{l}\alpha_i\alpha_j y_i y_j K(x_i, x_j) \qquad (10)$$

As stated in (Vapnik, 1998), among different kernel functions ($K(x_i, x_j)$), the kernel that minimizes 8 will yield the best SVM for the binary case.

In the previous papers (Hsu & Lin, ; Platt et al., 2000), a constant kernel function was used in their experiments for the entire multi-class problem. However, if the classes do not have similar structure, using only one kernel function may not be the best or it may not work for every binary classification. Thus, for best results, each binary classification has to be considered as an individual problem, and the best kernel should be chosen for each classifier. In this paper, we utilize an adaptive approach, which selects the best kernel for each SVM classifiers.

6. Experimental Results

We evaluate the performance of DB2 based on accuracy, training and testing times. We then compare the results with one-against-one, one-against-rest and DAGSVM. While in Table 2 we keep the kernel function and its parameter(s) constant, in Table 3 we

Table 2. Accuracies

	DB2		DAGSVM		One-against-One		One-against-rest	
	Rate	(C, δ)	Rate	(C, δ)	Rate	(C, δ)	Rate	(C, δ)
Glass	73.5	2^{11},2^{1}	**73.8**	2^{10},2^{-3}	72.0	2^{9},2^{-3}	71.9	2^{9},2^{-1}
Vowel	**99.2**	2^{10},2^{1}	**99.2**	Inf,2^{1}	99.0	2^{10},2^{0}	99.0	2^{10},2^{0}
HRCT	84.8	2^{11},2^{2}	82.4	2^{10},2^{3}	82.4	2^{11},2^{3}	**91.2**	2^{11},2^{2}
Modis	**70.1**	2^{10},2^{2}	69.7	2^{12},2^{2}	66.2	2^{10},2^{2}	69.3	2^{10},2^{2}
SmallModis	96.0	2^{10},2^{2}	**98.2**	2^{12},2^{3}	95.1	2^{10},2^{1}	96.5	2^{10},2^{2}
Segment	96.4	2^{9},2^{0}	**96.6**	2^{1},2^{3}	**96.6**	2^{9},2^{1}	95.2	2^{10},2^{1}

Table 1. Data

	#Samples	# Features	# Classes
HRCT	500	108	5
Modis	31299	169	15
SmallModis	5658	169	4
Glass	214	13	6
Vowel	528	10	11
Segment	2310	19	7

present the results for an adaptive approach. In our experiments, we tested algorithms with varying parameters at each step and observed the difference in accuracy. We determined the best kernel function and related parameters by running experiments on a validation set that is different from the test data. We present the experimental results in section 6.

We test the algorithms on six different data sets whose properties are provided in Table 1. Glass, vowel and segment are data sets from the UCI repository (Blake & Merz, 1998). HRCT data consists of high resolution computed tomography images of the lungs (Dy & Brodley, 2000). The classes represent various lung diseases.

Modis data is prepared by using the satellite images of the earth surface and consists of fifteen different classes representing fifteen different regions. Each region consists of various subregions. While selecting the test set, we picked all the samples from the subregions that are excluded from the training set. The Modis data has an imbalanced distribution. The number of samples in each class ranges from 261 to 6493.

We expect that if the problem consists of some small classes and some relatively large classes, then DB2 should be faster in the training phase. That is why, in order to illustrate this we also prepared a subproblem (SmallModis), using four classes from the Modis data. SmallModis has a skewed class distribution with a large class of 4502 samples and three smaller classes with 261, 411 and 466 samples.

6.1. Accuracy Comparison

In order to come up with more representative accuracy performances, we divided the large data sets (Modis, SmallModis and Segment) into three parts: The first part of data for training, the second one as a validation set to find the kernels and corresponding parameter(s) and the last part is used as testing data. For the data that has few samples, we used ten-fold-cross validation. We selected the best kernels among linear, polynomial and radial basis functions. For polynomial kernel parameters (δ), we limited our experiments from two through five and for RBF (δ) from 2^{-3} through 2^{5}. Another variable, which has a role in the accuracy of SVMs is the cost parameter (C). We repeated our experiment for various C values ranging from 2^{8} through 2^{12} and infinity.

We applied the Maxwins algorithm for combining the classifiers in one-against-one. We select the class giving the highest output value as the winner for one-against-rest. In case of an even voting or more than one class giving the highest value, we simply select the one with the lower index.

We show the results for DB2 and the third method presented in section 2.1, where we divided the classes into two subsets minimizing the difference of the number of data samples for each subset. Methods 1 or 2 gave similar accuracy performances.

Table 2 presents the results of the experiments, when the standard way of using a single kernel is applied. The best accuracy performances among the various multi-class approaches for each data are highlighted. For the HRCT and MODIS data, the polynomial kernel gave the best result for every algorithm. The radial basis function was the best kernel for the rest of the data sets. We also provide the corresponding cost parameter (C) and δ values in the table.

We believe that an adaptive approach should be utilized in any multi-class approach as pointed out in section 5 (i.e., the best kernel should be used for each classifier). Table 3 presents the results for adaptive DB2,

Table 3. Accuracies for Adaptive Kernels

	DB2 Rate	DAGSVM Rate	One-against-One Rate	One-against-rest Rate
Glass	**80.2**	79.3	76.6	75.1
Vowel	**99.2**	**99.2**	99.0	**99.2**
HRCT	92.2	86.4	83.7	**92.3**
Modis	70.4	**70.8**	68.2	70.1
SmallModis	**98.5**	**98.5**	98.1	97.0
Segment	96.4	**97.5**	**97.5**	96.2

and the adaptive versions for the other multi-class methods. As expected, the adaptive versions gave better accuracies for all the data sets. We observed that for Glass and HRCT data sets, the adaptive approach improved the accuracies significantly. On the other hand, for easily separable data sets or the ones that consists of similar structures, the adaptive version did not provide any significant improvement.

From Table 2, we can say that the four non-adaptive methods perform similarly in terms of accuracy. For the HRCT data, one-against-rest seems to be preferable. For the rest of the data sets, none of the four algorithms performed significantly better than the other. From Table 3, we observe that adaptive DB2 has a comparable performance with the best adaptive method for each data in this experiment. In the next subsection, we present a comparison of the algorithms in terms of speed.

6.2. Time Comparison

We ran the experiments on an UltraSparc-III cpu with 750 MHz clock frequency and 2GB RAM and the algorithms were implemented in matlab.

While testing the accuracies in the previous experiments, we also measured the CPU time consumed by each classifier and the results of the measurements are presented in Table 4. For the small data sets where ten fold cross validation is applied, we present the average of the total time spent for the experiment.

As seen from the results, DB2 is the fastest algorithm in most of the cases, with respect to testing time. Note that, we used the third criterion given in section 2.1, which has an important role in the speed of DB2. With this criterion, the larger classes are separated from the others in the earlier levels of the tree constructed by DB2. The smaller classes are left for the later levels, which makes it faster to train. Moreover, since most of the data to be tested comes from the larger classes, they are predicted in the earlier levels if no error occurs. Thus, most of the data is classified using less number of decision nodes, which speeds up the testing process. For instance, in the testing phase of DB2, the worst case happens when we apply $N-1$ nodes to the test data, and the best case occurs when the testing data can be predicted at the very first level. If we separate the largest class from the others at the first level, most of the testing data can be predicted using only one decision function of DB2. In other words, only one binary SVM is enough for most of the testing data.

In HRCT and SmallModis data, there exist smaller classes and a relatively larger class. As mentioned in section 3, in such cases where data is skewed or unbalanced, DB2 becomes preferable when we consider training time.

On the other hand, if there are no skewed classes within the problem, DB2 may lose its advantage. In case, where data is evenly distributed as in the segment data set, DAGSVM can be faster depending on the δ value of the problem. As described in section 3, if $\delta = 2$, DAGSVM, one-against-one and DB2 have the same time complexity in training but for $\delta > 2$, DAGSVM and one-against-one become faster in training than other algorithms.

To understand better the time complexity of the various multi-class methods with respect to the number of instances, we ran experiments with increasing numbers of segment data samples. Using the same parameters ($\delta = 2$, C = Inf.), we measured the CPU time for the testing and training phases. In order to obtain homogeneous data sets with different numbers of samples, we started with the first 100 samples of the segment data and incremented it by 100 at each measurement. We took 90% of the data as training and the rest as testing.

Figure 3 and 4 display the plots for the training and testing time respectively. Results show that one-against-one and DAGSVM are the fastest methods in training, followed closely by DB2. One-against-rest is significantly slower than the other three. When we consider the testing time, we observe that DB2 and

Table 4. CPU Time (in Seconds)

	DB2		DAGSVM		One-against-One		One-against-rest	
	Train	Test	Train	Test	Train	Test	Train	Test
Glass	48.7	**2.1**	32.3	2.6	**31.0**	6.3	142.6	6.8
Vowel	383.6	**15.7**	**198.3**	18.0	212.9	97.9	2881.1	82.4
HRCT	**330.6**	**18.2**	378.6	33.6	376.3	54.4	1336.1	83.5
SmallModis	**3658.2**	**674.3**	4236.5	982.6	4165.2	1876.5	23256.1	2958.5
Modis	236700	**4203**	35214	9590	**34927**	19060	973632	12008
Segment	1934.6	441.0	1549.4	**417.8**	**1536.8**	1227.1	9470.6	2036.4

DAGSVM are substantially faster than the other two.

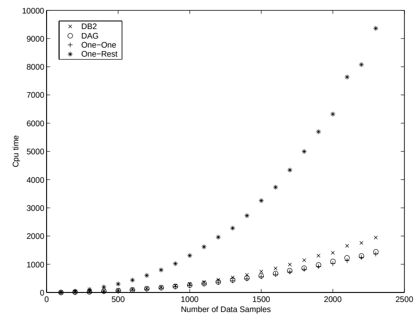

Figure 3. Training Time

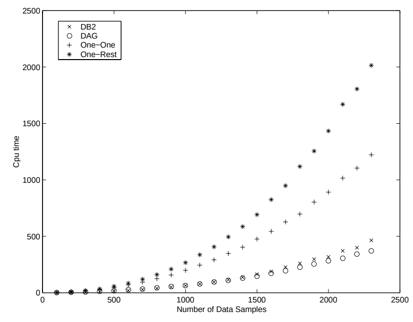

Figure 4. Testing Time

7. Conclusions and Future Work

We have introduced a new method as a solution to multi-class problems. DB2 has a flexible tree structure and can be adjusted for different types of multi-class problems. Benefiting from the tree structure, we were able to present a generalization and time complexity analysis. Our experiments show that for typical cases, DB2 can be trained as fast as one-against-one algorithms. Looking at the results, we can conclude that DB2 is always faster than one-against-one and one-against-rest algorithms in terms of testing time. Furthermore, it is faster than DAGSVMs when the data is unbalanced. For other data sets, DB2's speed is close to DAGSVM's. In conclusion, we can say that DB2 is an alternative to other multi-class methods with comparable accuracy performance and is preferable with respect to speed, depending on the problem.

We also suggest that determining the best kernel functions and parameters for every classifier within the multi-class architecture can improve the accuracy significantly, depending on the distribution of the data. Our experimental results confirmed that indeed an adaptive approach significantly improves the classification accuracy.

As an extension to DB2, we can combine other existing multi-class methods with DB2 at different levels of DB2 and produce a hybrid structure. For instance, up to some level we can split the data into two and then apply DAGSVM for the rest of the classes. Furthermore, if we split one class out at every stage, we come up with an algorithm that is very similar to one-against-rest but faster than that. The idea is to combine the strength of each multi-class method.

Another direction is to explore the benefits of using different set of features at each level of the hierarchy, similar to Dumais and Chen (2000). At each node different features may be more relevant. Intuitively, we expect that the idea can improve the accuracy for problems with natural hierarchies. Moreover, we expect that an adaptive approach would gain more importance when we allow the feature space to change at each node.

The methods (1 & 2) provided in section 2.1 summarize each class using first-order moment statistics. We can take advantage of second-order moments summaries and DB2 by optimizing discriminant analysis functions such as $\mathrm{tr}(S_w^{-1}S_b)$ where S_w is the within-class-scatter and S_b is the between-class-scatter matrices (Fukunaga, 1990). One may also search for the best grouping by incorporating the kernel functions in the criterion function. Determining the best method for grouping the classes would be an interesting topic for future work.

Acknowledgments

The authors wish to thank Mark Friedl from Boston University for the Modis data. This research was partially supported by Mercury Computer Systems, the NSF funded CenSSIS (Center for Subsurface Sensing and Imaging Systems), and NSF Grant No. IIS-0347532.

References

Bennett, K. P., Cristianini, N., Shawe-Taylor, J., & Wu, D. (2000). Enlarging the margins in perceptron decision trees. *Machine Learning*, *41*, 295–313.

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Crammer, K., & Singer, Y. (2000). On the learnability and design of output codes for multiclass problems. *Computational Learing Theory* (pp. 35–46).

Dumais, S. T., & Chen, H. (2000). Hierarchical classification of Web content. *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval* (pp. 256–263). Athens, GR: ACM Press, New York, US.

Dy, J. G., & Brodley, C. E. (2000). Visualization and interactive feature selection for unsupervised data. *Knowledge Discovery and Data Mining* (pp. 360–364).

Forgy, E. (1965). Cluster analysis of multivariate data: Efficiency vs interpretability of classifications. *Biometrics*, *21*, 768–780.

Friedman, J. (1996). *Another approach to polychotomous classifcation* (Technical Report). Stanford University, Department of Statistics.

Fukunaga, k. (1990). *Introduction to statistical pattern recognition*. Boston, MA: Academic Press. 2 edition.

Hsu, C., & Lin, C. A comparison of methods for multiclass support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2001. 19.

Knerr, S., Personnaz, L., & Dreyfus, G. (1990). Single-layer learning revisited: A stepwise procedure for building and training a neural network. *Neurocomputing: Algorithms, Architectures and Applications, NATO ASI Series*. Springer.

Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report 98-14, Microsoft Research, Redmond, Washington, April 1998. http://www.research.microsoft.com/ jplatt/smo.html.

Platt, J., Cristianini, N., & Shawe-Taylor, J. (2000). Large margin dags for multiclass classification. *Advances in Neural Information Processing Systems 12* (pp. 547–553).

Vapnik, V. (1995). *The nature of statistical learning theory*. New York: Springer.

Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley.

Weston, J., & Watkins, C. (1999). Support vector machines for multiclass pattern recognition. *Proceedings of the Seventh European Symposium On Artificial Neural Networks*.