Scaling multi-class Support Vector Machines using inter-class confusion

Shantanu Godbole
IIT Bombay
shantanu@it.iitb.ac.in

Sunita Sarawagi
IIT Bombay
sunita@it.iitb.ac.in

Soumen Chakrabarti
IIT Bombay
soumen@cse.iitb.ac.in

ABSTRACT

Support vector machines (SVMs) excel at two-class discriminative learning problems. They often outperform generative classifiers, especially those that use inaccurate generative models, such as the naïve Bayes (NB) classifier. On the other hand, generative classifiers have no trouble in handling an arbitrary number of classes efficiently, and NB classifiers train much faster than SVMs owing to their extreme simplicity. In contrast, SVMs handle multi-class problems by learning redundant yes/no (one-vs-others) classifiers for each class, further worsening the performance gap. We propose a new technique for multi-way classification which exploits the accuracy of SVMs and the speed of NB classifiers. We first use a NB classifier to quickly compute a confusion matrix, which is used to reduce the number and complexity of the two-class SVMs that are built in the second stage. During testing, we first get the prediction of a NB classifier and use that to selectively apply only a subset of the two-class SVMs. On standard benchmarks, our algorithm is 3 to 6 times faster than SVMs and yet matches or even exceeds their accuracy.

1. INTRODUCTION

Support Vector Machines (SVMs) [14] are a kind of *discriminative* classifier which have shown superb performance for classifying text and other data. They are accurate, robust, and quick to apply to test instances. Inducing a linear SVM over training data $\{(\vec{x_i}, y_i), i = 1, \ldots, n\}, x_i \in \mathbb{R}^m, y_i \in \{-1, 1\}$ involves estimating a vector $\vec{w}$ and a scalar b to maximize the distance of any training point from the hyperplane defined by $\vec{w} \cdot \vec{x} + b$; this can be written as:

$$\begin{array}{ll} \text{Minimize} & \frac{1}{2}\,\vec{w} \cdot \vec{w} \qquad (= \frac{1}{2}\|\vec{w}\|^2) \\ \text{subject to} & y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1 \quad \forall i = 1, \ldots, n \end{array} \tag{1}$$

The distance of any training point from the optimized hyperplane (called the *margin*) will be at least $1/\|w\|$. Notice that there can be only two class labels. This holds for non-linear SVMs as well.

The elegant theory behind the use of large-margin hyperplanes to separate two classes cannot be extended easily to separate N mutually exclusive classes. A number of methods have been proposed for reducing a multi-class problem to a collection of two-class problems and combining their predictions in various ways [1, 12, 7]. The most popular amongst these is the "one-vs-others" approach where, for each of the N classes, we construct a one-vs-others SVM which makes a yes/no judgment for that class alone. Given a test instance, some of these N SVMs will say yes, and the rest will say no. The winning SVM is the one which says yes, and whose decision hyperplane is farthest from the test instance amongst all competing SVMs.

The one-vs-others technique requires each of the N classifiers to be built on the *entire* training data, causing each document to be processed N times. Each test instance has to be evaluated w.r.t. each SVM as well. Apart from the efficiency issue, it is unclear that a comparison of the discriminant functions of *different* classification problems is meaningful in any way, although this technique seems to work reasonably well in practice.

Another technique is to construct SVMs between all possible pairs of classes [10]. During testing, each of the $\frac{N(N-1)}{2}$ classifiers votes for one class. The winning class is the one with the largest number of accumulated votes. In this technique, even though the number of training instances per classifier is limited, the number of classifiers is quadratic in the number of classes and each document gets processed $(N-1)$ times. A number of other proposed methods [5, 3, 1] fall somewhere in between these two methods. We will discuss these in Section 4. The one-vs-others method is the most widely used in practice and found to offer high accuracy [7]. We will use this approach when using multi-class SVMs in our experiments.

Regardless of specific details, these ad-hoc techniques become impractical with a large number of classes, especially because SVMs, while accurate and quick to apply, are not the fastest learners to train: on an n-instance problem the time taken by recent, clever implementations ranges from $n^{1.7}$ to $n^{2.1}$ [8, 4]. Together, these factors make it quite difficult to scale up SVM-based systems to large Web directories like the Open Directory Project (`http://dmoz.org/`) and Yahoo! (`http://www.yahoo.com/`), which have tens of thousands of classes and millions of documents.

Generative classifiers, in contrast, are essentially independent of the number of classes as far as training time is concerned. A popular generative classifier for text data is the naïve Bayes (NB) classifier. NB classifiers trivially scale with the number of classes as they process each document

SIGKDD '02 Edmonton, Alberta, Canada

only once independent of the number of classes. Also, NB classifiers train much faster than SVMs owing to their extreme simplicity. On the other hand, in terms of accuracy, the linear SVM has decisively outperformed the NB classifier owing to the latter's high bias in assuming attribute independence.

In this paper, we present a method that tries to achieve the best of both worlds: scalability of NB classifiers w.r.t. number of classes and accuracy of SVMs. In the first stage we use the fast multi-class NB classifier to compute a confusion matrix, which is used to reduce the number and complexity of two-class SVMs that are built in the second stage using the one-vs-others approach. During testing, we first get the prediction of a NB classifier and use that to selectively apply only a subset of the two-class SVMs, as indicated by the confusion matrix. On standard benchmarks, our algorithm is 3 to 6 times faster than multi-class SVMs, and has superior scalability in terms of memory requirements and training set size. In terms of accuracy, the method is 3% better than NB classifiers and comparable or superior to SVMs.

Our proposed algorithm is very simple to understand and requires negligible coding. It can be of substantial utility while dealing with very large classifiers like those that would be required for Web directories.

2. OUR APPROACH

The genesis of our approach lies in the class relationships derived from a confusion matrix, easily generated from a fast classifier like NB. This can be obtained using a held-out validation dataset.

Classname		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
alt.atheism	1	251	6	1	3	32	1	1	2	1	2	0	0	0	0	0	0	0	0	0	0
soc.religion.christian	2	9	277	0	1	6	0	0	1	0	0	0	0	0	1	2	2	0	0	0	1
sci.space	3	3	1	273	1	0	1	2	0	1	1	9	0	0	1	2	3	0	0	1	1
talk.politics.misc	4	2	0	3	213	24	3	0	17	3	0	0	0	0	0	0	1	0	1	33	0
talk.religion.misc	5	88	36	2	23	132	0	1	0	0	0	0	0	0	0	0	2	0	1	15	0
rec.autos	6	0	0	0	3	1	272	0	0	0	7	1	2	1	6	4	1	0	0	2	0
comp.windows.x	7	1	1	2	1	0	1	246	0	2	2	30	5	3	1	1	2	1	1	0	0
talk.politics.mideast	8	0	3	1	18	0	0	0	275	0	1	0	0	0	0	0	0	0	1	1	0
sci.crypt	9	1	0	1	2	1	0	3	0	284	0	3	0	1	0	0	1	0	0	3	0
rec.motorcycles	10	0	0	0	1	0	4	1	0	0	286	1	2	0	1	2	1	0	0	1	0
comp.graphics	11	0	1	2	1	1	0	10	1	2	0	243	23	7	3	3	3	0	0	0	0
comp.sys.ibm.pc.hardware	12	0	0	0	0	0	2	7	0	1	0	5	243	23	12	3	1	3	0	0	0
comp.sys.mac.hardware	13	0	0	1	1	0	2	1	0	0	0	7	10	260	8	9	1	0	0	0	0
sci.electronics	14	1	0	1	0	1	5	2	0	2	0	7	13	13	245	6	3	0	1	0	0
misc.forsale	15	0	1	4	2	0	12	1	0	0	4	1	19	10	8	233	1	0	1	1	2
sci.med	16	0	1	5	0	1	1	0	0	0	1	2	0	2	7	2	275	0	1	1	1
comp.os.mswindows.misc	17	1	0	2	0	1	1	58	1	3	0	38	71	17	3	6	0	97	1	0	0
rec.sport.baseball	18	2	1	1	0	0	0	0	0	0	0	4	0	0	0	1	1	0	282	1	7
talk.politics.guns	19	0	0	0	9	5	1	0	0	1	0	0	0	0	1	0	0	1	1	281	0
rec.sport.hockey	20	0	1	0	0	0	1	0	0	0	2	0	0	1	1	0	0	0	3	0	291

Figure 1: 20-newsgroups confusion matrix

For example, in Figure 1 we show an example of a confusion matrix built on the 20-newsgroup dataset (details in Section 3.1). The rows show actual classes and the columns show predicted classes.

The matrix clearly shows that different classes have different degrees of confusion with other classes. Some classes, like `rec.sport.hockey`, are well separated from the rest, whereas others like `comp.os.ms-windows.misc` are easily confused with others. The mis-classifications of a class are usually limited to a small subset of classes. In fact, in most cases, the rows and columns of a matrix can be rearranged manually as shown in Figure 2 so as to reveal clusters of classes that confuse with each other. These appear as blocks along the diagonal of the confusion matrix. Not surprisingly, in many cases, these clusters are formed of classes whose names can be immediately recognized as forming natural hierarchies. The confusion matrix provides a domain-independent method of deriving this relationship.

For automating this re-organization of classes into clusters of similar classes, we use the technique used in [6] to automatically generate topic hierarchies from a given flat set of classes. Each class is represented by a row in the confusion matrix. For each class, it's respective row is converted to a normalized N dimensional vector that denotes how much the class confuses with other classes. We then use a distance measure like the Euclidean L_n or the KL-distance measure to compute distance between the classes. These distances are used to cluster the classes using a hierarchical agglomerative clustering (HAC) algorithm. The output of HAC is a dendrogram that we analyze to determine the clusters that provide the maximum inter-class separation. The dendrogram is scanned bottom-up to find the distances at which successive clusters get merged. We clip the dendrogram at the point where the cluster merge distances begin increasing sharply. The number of clusters left after clipping form the clusters of our two-level hierarchy. For the 20-newsgroups dataset, this method gives clusters very similar to those in Figure 2.

Classname		1	2	5	19	8	4	11	17	12	13	7	15	14	6	10	18	20	9	16	3
alt.atheism	1	251	6	32	0	2	3	0	0	0	0	1	0	0	1	2	0	0	1	0	1
soc.religion.christian	2	9	277	6	0	1	1	0	0	0	0	0	2	1	0	0	0	1	0	2	0
talk.religion.misc	5	88	36	132	15	0	23	0	0	0	0	1	0	0	0	0	1	0	0	2	2
talk.politics.guns	19	0	0	5	281	0	9	0	1	0	0	0	0	1	1	0	1	0	1	0	0
talk.politics.mideast	8	0	3	0	1	275	18	0	0	0	0	0	0	0	0	1	1	0	0	0	1
talk.politics.misc	4	2	0	24	33	17	213	0	0	0	0	0	0	0	3	0	1	0	3	1	3
comp.graphics	11	0	1	1	0	1	1	243	0	23	7	10	3	3	0	0	0	0	2	3	2
comp.os.mswindows.misc	17	1	0	1	0	1	0	38	97	71	17	58	6	3	1	0	1	0	3	0	2
comp.sys.ibm.pc.hardware	12	0	0	0	0	0	0	5	3	243	23	7	3	12	2	0	0	0	1	1	0
comp.sys.mac.hardware	13	0	0	0	0	0	1	7	0	10	260	1	9	8	2	0	0	0	0	1	1
comp.windows.x	7	1	1	0	0	0	1	30	1	5	3	246	1	1	1	2	1	0	2	2	2
misc.forsale	15	0	1	0	1	0	2	1	0	19	10	1	233	8	12	4	1	2	0	1	4
sci.electronics	14	1	0	1	0	0	0	7	0	13	13	2	6	245	5	0	1	0	2	3	1
rec.autos	6	0	0	1	2	0	3	1	0	2	1	0	4	6	272	7	0	0	0	1	0
rec.motorcycles	10	0	0	0	1	0	1	1	0	2	0	1	2	1	4	286	0	0	0	1	0
rec.sport.baseball	18	2	1	0	1	0	0	4	0	0	0	0	1	0	0	0	282	7	0	1	1
rec.sport.hockey	20	0	1	0	0	0	0	0	0	0	0	0	1	1	1	2	3	291	0	0	0
sci.crypt	9	1	0	1	3	0	2	3	0	0	1	3	0	0	0	0	0	0	284	1	1
sci.med	16	0	1	1	1	0	0	2	0	0	2	0	2	7	1	1	1	1	0	275	5
sci.space	3	3	1	0	1	0	1	9	0	0	0	2	2	1	1	1	0	1	1	3	273

Figure 2: 20-newsgroups re-organized confusion matrix.

2.1 Hierarchical Approach

We propose to exploit the clustering of classes to prune the number and complexity of two-class classifiers needed for multi-class SVMs. An obvious approach is to arrange the clusters in a two-level tree hierarchy and train a classifier at each internal node. If we restrict to NB classifiers, Mitchell [11] has shown that if the same feature space were used for all the classifiers and no smoothing is done, the accuracy would be the *same* as that of a flat classifier. In practice, each classifier has to deal with a more easily separable problem, and can use a independently optimized feature set; this should lead to slight improvements in accuracy, apart from the gain in training and testing speed. We propose to use a combination of NB and SVM classifiers at the two levels.

We first build a top-level classifier to discriminate amongst the top-level clusters of labels, called the Level 1 (L1) classifier. This top-level classifier could be either a NB or SVM classifier. Even with SVM, the training time will be smaller since the number of classes is reduced, although each two-

Method	Accuracy (in %)
MCNB	85.27
MCSVM	89.66
NB-L1	93.56
SVM-L1	95.39
NB-L2 with NB-L1	89.01
NB-L2 with Perfect-L1	88.41
SVM-L2 with NB-L1	92.04
SVM-L2 with Perfect-L1	91.65

Table 1: Level-wise comparisons for 20newsgroups data

class SVM will still need all documents. At the second-level (L2) we build multi-class SVMs within each cluster of classes. The total number of SVMs at the second level will be close to N but the number of classes per SVM is significantly reduced.

Each L2 classifier can concentrate on a smaller set of classes that confuse with each other. For a generative classifier like NB, we expect this to result in better feature selection and thus enable finer distinctions amongst the confusing classes [2]. For SVMs, the spread of the negative ('others') class is reduced, which we expect will make separability easier and/or better.

2.1.1 Evaluation of the hierarchical approach

We evaluate the accuracy and training time for solving the multi-class problem using a two-level hierarchy. We compare four methods: Flat multi-class NB classifiers (MCNB), flat multi-class SVMs (MCSVM) using the one-vs-others approach, NB classifiers at both the levels (L1 and L2) of the hierarchy (Hier-NB) and NB classifier at L1 followed by SVMs at L2 (Hier-SVM).

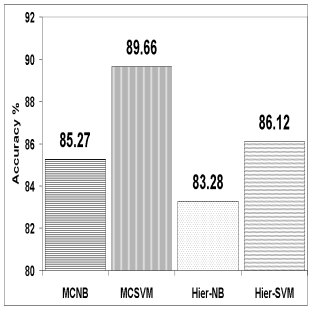

Figure 3: Accuracy of different hierarchical methods for 20NG

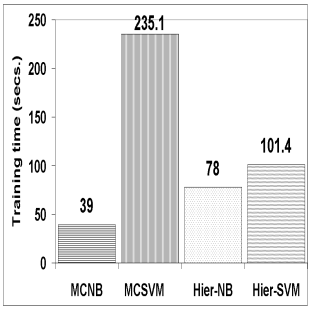

Figure 4: Training time of different hierarchical methods for 20NG

We notice from Figures 3 and 4 that while the training times for Hier-NB and Hier-SVM are lower than for MCSVM, the accuracy is also reduced. The accuracy for Hier-NB is even lower than that of MCNB. Hier-SVM has lower accuracy than MCSVM though it is slightly better than MCNB. Also, training time of Hier-SVM is less than half that of MCSVM.

We show in Table 1, a comparison of accuracy of the two levels separately for both NB and SVM classifiers. For L2, we show two kinds of accuracy, first on documents that get correctly classified to their correct group by L1, and second the absolute accuracy of L2 assuming a perfect L1 classifier. As expected, all the L1 and L2 classifiers are individually more accurate than the original flat classifier, as noted in [2] and as we expected in Section 2.1. Even though NB-L2 has an accuracy of 89.01%, combining with the NB-L1 accuracy of 93.56% leaves us with a resultant accuracy of Hier-NB of 83.28%. Although NB-L1 and NB-L2 are both individually better than MCNB (85.27%), we see that compounding of classification errors leaves Hier-NB worse off than MCNB. Similarly, SVM-L2 with a NB classifier at L1 (Hier-SVM) has an accuracy of 92.04%. The accuracy of Hier-SVM still drops down to 86.12% which is slightly better than MCNB, but is worse than the MCSVM accuracy of 89.66%. If we replace NB-L1 with SVM-L1 having 95.39% accuracy, the overall accuracy for both Hier-NB and Hier-SVM improves slightly, but the training time gets worse.

There is no conclusive comparison in [2, 9] which decisively states whether a hierarchical classification scheme is better than a flat one for NB classifiers. These previous studies have either restricted the number of features at each node in the hierarchy or have tried to equalize the number across the flat and hierarchical schemes. It is not clear whether by attempting to equalize the number of features, one of the classification schemes is getting compromised.

The main reason for the low accuracy of the hierarchical approaches is the compounding of errors at the two levels. Even though the accuracy at both levels is higher than that of a flat classifier, the product of their accuracies falls short of the accuracy of a flat classifier. Increasing the levels beyond two is expected to worsen this compounding effect.

This led us to design a new algorithm GraphSVM, that attempts to ensure that the first-stage classification will make the overall process fast, but inaccuracy of the first-stage classifier will not jeopardize overall accuracy.

2.2 The GraphSVM algorithm

In this algorithm, we represent class confusion in a more general way using a graph, which may connect a class with any other class, instead of restricting confusion to a hierarchy of disjoint groups as in the previous approach.

As in the previous approach, we start with the confusion matrix obtained by a fast multi-class NB classifier M_1. For each class i, we find the set of classes $F(i)$ such that more than a threshold (t) percentage of documents from each class in $F(i)$ gets mis-classified as class i. For example, for the confusion matrix in Figure 1, we find that for the class `alt.atheism` and with a threshold of 3%, $F($`alt.atheism`$) =$ {`talk.religion.misc`, `soc.religion.christian`}.

Next, for each node i with non-empty $F(i)$, we train a multi-class classifier $M_2(i)$ to distinguish amongst the classes in $\{i\} \cup F(i)$. These classifiers are constructed using a more accurate and possibly slower method like SVMs.

During testing, we first classify a document d using M_1. If the predicted class for d is i, we feed it to $M_2(i)$, if any, and get a refined prediction j.

In the above example, when a test instance is predicted as `alt.atheism` by M_1, we get the prediction refined by a one-vs-others SVM, $M_2(i)$, between the classes `alt.atheism`, `talk.religion.misc` and `soc.religion.christian`. The prediction of $M_2(i)$ is returned as the final answer for the test instance.

2.2.1 Discussion of the algorithm

GraphSVM partitions a classification task between NB and SVMs such that SVMs are only invoked on small subsets of classes that get mis-classified by the NB classifiers.

We can claim that GraphSVM will not be worse than MCNB provided that the training data is representative of the test data. GraphSVM can choose not to use any second stage SVM refinements in the rare case where the SVM classifier is found to be worse than a NB classifier on a validation dataset. Compared to MCSVM, the main reason GraphSVM may be worse is high positive values of the threshold t. We can always decrease the threshold, at the expense of increasing the training time, to match MCSVM accuracy as shown in Section 3.5. In most cases, we expect the strengths of NB and SVMs to combine to give performance better than both individually.

The main property that we rely on for accuracy is that the set of classes that a class confuses with, using a NB classifier, should be the same for the training and test set. The relative distribution of the confusion matrix is not required to remain unchanged, provided entries that were previously below the threshold t do not suddenly increase beyond it.

The benefit of GraphSVM will be greatest when the mis-classifications of the first stage are spread across a small number of classes. The worst case is when mis-classifications of a class are uniformly distributed over many classes. In this case, the algorithm will reduce to multi-class SVMs. In most practical datasets that is rarely the case.

3. EXPERIMENTAL EVALUATION

Text classification involves dealing with tens of thousands of features and documents. NB classifiers have proved to be very fast and scalable and show a moderate accuracy, whereas the SVM variants are the most accurate, but the slowest to train. Our aim is to scale up these expensive SVMs using the proposed GraphSVM algorithm. Hence we compare GraphSVM only with the fast NB classifiers and the accurate SVM classifiers.

In this section we present an evaluation of the proposed GraphSVM algorithm. We compare this algorithm to multi-class NB (MCNB), multi-class SVMs (MCSVM) and the hierarchical approach with a NB classifier at L1 and SVMs at L2 (Hier-SVM). We compare the algorithm on accuracy, training time and scalability w.r.t. size of the training set.

3.1 Datasets

20-newsgroups The 20-newsgroups (20NG) dataset[1] is a collection of 18,828 news wire articles from 20 Usenet groups. The older version of the dataset had 1,000 articles in each group, but the newer version we use is without duplicates and with most headers removed. This dataset is not pre-processed into training and testing sets. We randomly chose 70% of the documents for training and the remaining 30% for testing. The corpus contained around 75,000 words. Features were selected using mutual information. All words were stemmed using a Porter stemmer [13], all HTML tags were skipped and all header fields except subject and organization of the posted article were ignored.

Reuters-21578 The Reuters-21578 Text Categorization Test collection[2] is a standard text categorization benchmark. It contains 135 classes. We chose only those 60 classes which have more than 10 training documents. This resulted in 8819 training documents and 1887 test documents. XML tags were ignored and the words were stemmed with a Porter stemmer [13]. We used the standard Mod-Apte train-test split. Training instances with multiple class assignments were considered once in every assigned class. We ignored multi-class test instances because we wanted to see if confusion amongst classes can be resolved by using the proposed algorithm.

All experiments were performed on a 1.4GHz P4 machine with 512MB RAM, running Linux. *Rainbow*[3] was used for feature selection, text processing and experiments involving NB classifiers. *SVMLight*[4] was used for all experiments involving SVMs.

3.2 Overall comparison

In Figures 5 and 6 we show the accuracy and training time for the four methods on the 20NG and Reuters datasets.

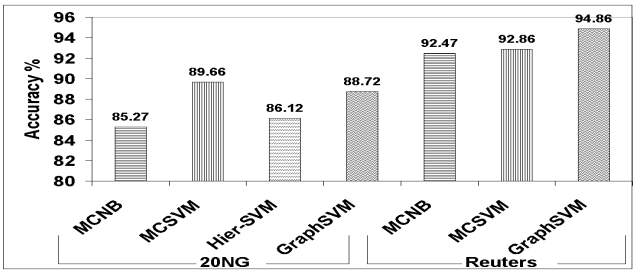

Figure 5: Accuracy comparison for all methods for both datasets

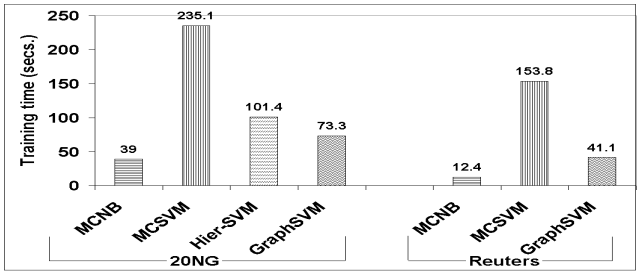

Figure 6: Training time comparison for all methods for both datasets

In Figure 5, for the 20NG dataset, we observe that the accuracy of GraphSVM at 88.72% is slightly smaller than the MCSVM accuracy of 89.66%. The accuracy of GraphSVM is higher than the previous Hier-SVM accuracy of 86.12%. For the Reuters dataset, GraphSVM has the highest accuracy of 94.86% while MCSVM and MCNB have accuracies of 92.86% and 92.47% respectively.

From Figure 6 we observe that GraphSVM has the fastest training time among the approaches involving SVMs, and for both the datasets, is more than a factor of 3 faster than MCSVM. As expected, MCSVM is the slowest to train.

3.3 Scalability with number of classes

We evaluated the training time of the different approaches with increasing number of classes. We started with 5 randomly picked classes, and added 5 randomly picked classes at a time for both the datasets. In Figures 7 and 8 we observe that the gap between the training time of GraphSVM and MCSVM increases as the number of classes is increased.

[1] http://www.ai.mit.edu/~jrennie/20_newsgroups/

[2] http://www.daviddlewis.com/resources/testcollections/reuters21578/

[3] http://www.cs.cmu.edu/~mccallum/bow/

[4] http://svmlight.joachims.org/

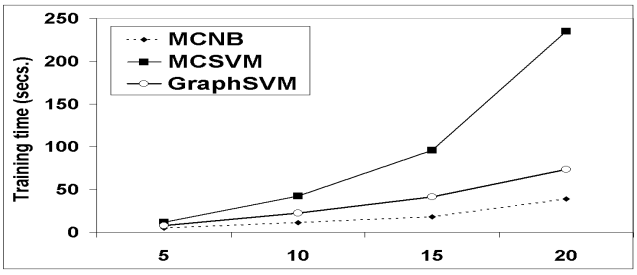

Figure 7: Training time vs. Number of classes for 20NG

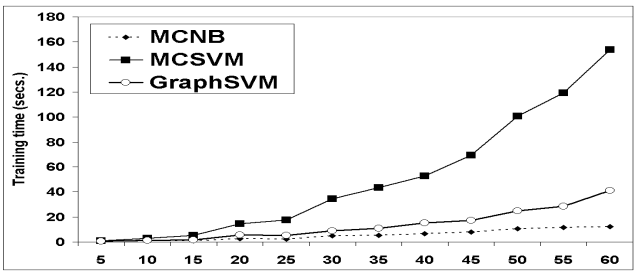

Figure 8: Training time vs. Number of classes for Reuters

Figures 9 and 10 show that in all cases GraphSVM continues to maintain the high accuracy vis-a-vis MCSVM and MCNB. For the 20NG dataset (Figure 9), GraphSVM maintains an accuracy of within 1% of MCSVM and for the Reuters dataset (Figure 10) GraphSVM is on an average, 3% better than MCSVM. We notice a large dip in Figure 10 for MCNB and MCSVM. The Reuters dataset is highly skewed in the distribution of instances per class. Figure 10 additionally shows the total number of test instances over which the micro-averaged accuracy values are reported. For 25 classes there are only 166 test instances. The 7% difference between GraphSVM and MCNB and MCSVM is due to only 11 additional instances correctly classified by GraphSVM. Since these 25 classes are thinly populated, most of the misclassifications seen in the confusion matrix are larger than the threshold and contribute to edges in the GraphSVM algorithm. These mis-classifications are corrected by the more focused SVMs in the GraphSVM method. The graph smoothens out after 30 classes when there are a larger number of instances and the accuracy of GraphSVM is consistently better.

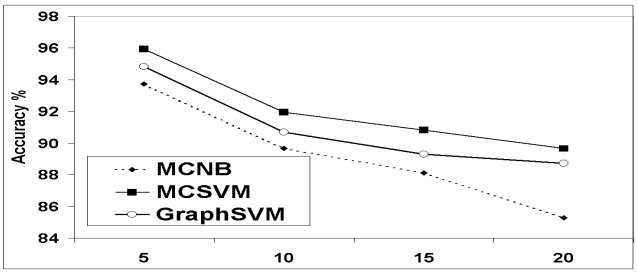

Figure 9: Accuracy vs. Number of classes for 20NG

3.4 Scalability with training set size

Training time: In Figure 11, the size of the training set was varied from 10% to 70% of the whole data, while keeping the relative train-test ratio constant at 70:30. We observe

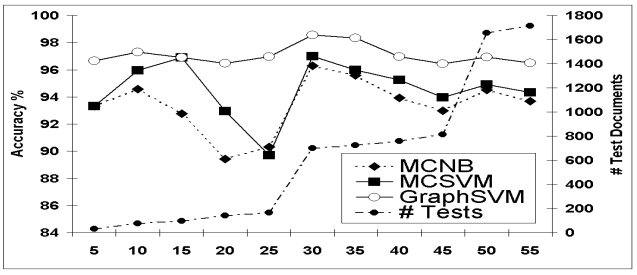

Figure 10: Accuracy vs. Number of classes for Reuters

that the training time of GraphSVM is nearly linear in the training set sizes, while for multi-class SVMs the training time increased super-linearly with training set size. This causes the gap between the two methods to become more prominent for larger datasets.

Accuracy: In Figure 11 we show the corresponding accuracy values against varying percentages of training set sizes for 20NG. We observe that as the accuracy of MCSVM increases with increasing number of training instances and GraphSVM closely tracks the increase and is always more accurate than MCNB.

Maximum memory: In Figure 11 the percentage of training documents is plotted against the maximum memory required to train any SVM model in the GraphSVM and MCSVM approaches. In both cases, multiple one-vs-others SVMs are learned, but the size and heterogeneity of the negative ('others') class varies largely leading to different memory requirements. In MCSVM, this negative class contains the entire dataset apart from the positive class, whereas it is greatly pruned in the GraphSVM approach. We notice that GraphSVM requires less than one-fourth the memory required by MCSVM.

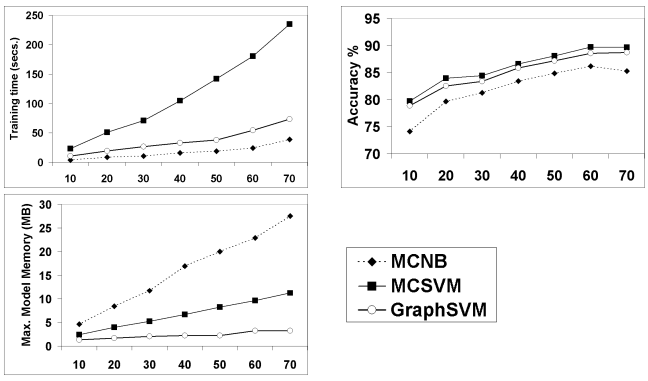

Figure 11: Training time, accuracy and maximum model memory with varying training set sizes for the 20NG dataset

3.5 Effect of the threshold parameter

An important parameter of GraphSVM is the threshold used to decide what SVMs to create in the second stage.

In Table 2 we show the accuracy and training time for different values of the threshold (t). We see that a threshold of 3% to 7% is appropriate for both these datasets because of the base accuracy of the NB classifier chosen to get the confusion matrix.

Method	Threshold t	Training time (secs)	Accuracy in %
Reuters			
MCSVM	-	158	92.86
GraphSVM	0.07	36	94.91
GraphSVM	0.05	41	94.86
GraphSVM	0.03	46	94.86
GraphSVM	0.01	65	95.33
20NG			
MCSVM	-	235	89.66
GraphSVM	0.07	63	87.92
GraphSVM	0.05	73	88.72
GraphSVM	0.03	73	88.52

Table 2: Accuracy and Performance

If the threshold value is kept unnecessarily high, we will hardly have any graph to construct, assuming the base classifier has a decent accuracy. On the other hand, even for a moderately accurate base classifier, a very low threshold (say 2% or less) will make a densely connected graph. In that case, GraphSVM will approach MCSVM.

As seen in Sections 3.2 and 3.3, GraphSVM is more accurate than MCSVM and MCNB for the Reuters dataset and also requires less training time. The scalability results for the Reuters dataset w.r.t. training set size are the same as that for the 20NG dataset in Section 3.4, viz. GraphSVM is highly scalable w.r.t. training time, accuracy and memory requirements as compared to MCSVM. The detailed results are omitted here for space constraints.

4. RELATED WORK

A general framework for solving multi-class problems using a collection of two-class problems is to associate it with a coding matrix M where each class is a row and each two-class classifier is a column [3, 1]. Each element a_{ij} is $+1$,-1 or 0, where $+1$ denotes the class i serving as a positive class in classifier j, -1 denotes i serving as a negative class in j and 0 denotes that class i does not participate in classifier j. The coding matrix for one-vs-others will have N columns with $+1$ on the diagonal and -1 in all other entries. For Max-Wins [10], the coding matrix will have $\binom{N}{2}$ columns and each column will have a single $+1$ and -1 and the rest of the entries 0. Error Correcting Output Codes (ECOC) classifiers are proposed in [3] where the $+1$s and -1s are chosen in such a way that the different rows are maximally separated. During testing, the outcome of each classifier is treated as a vector and compared with each row. The row to which it is closest is returned as the predicted class. Ghani [5] reports experiments with a number of ECOC classifiers to solve large multi-class text classification problems using NB. They report significant improvement in accuracy with the ECOC method on the Industry section dataset. Rennie and Rifkin [7] repeated similar experiment with Support Vector Machines as the base classifier. On the standard text benchmarks, they found that for SVMs the various ECOC classifiers did not provide any accuracy improvement over the simple one-vs-others method.

Platt et al [12] present a modification of the testing procedure of the Max-Wins algorithm which reduces the number of kernel evaluations during testing. They arrange the various two-class classifiers in a DAG structure, that is used to order the application of various two-class SVMs during testing. This method does not affect the training time and the accuracy is comparable.

5. CONCLUSION

We have described GraphSVM, an effective framework for extending discriminative classifiers like SVMs to handle data with a large number of classes, accurately and efficiently. GraphSVM estimates a measure of affinity between classes which depends upon the severity of confusion between these classes by a fast classifier like NB. This confusion indicates a clustering of classes which is used to limit the number and complexity of the SVMs that need to be trained for multi-class categorization. GraphSVM beats the accuracy of multi-class NB decisively even though it builds on a NB classifier. GraphSVM outperforms SVMs w.r.t. training time and memory requirements. It matches or even exceeds the accuracy of multi-class SVMs. GraphSVM is very simple to understand and requires negligible coding, but it can be of substantial utility while dealing with very large classifiers with tens of thousands of classes and millions of instances.

Acknowledgments

This work was funded by a research grant from the Ministry of Information Technology, India. We wish to thank Anand Janakiraman, Ravindra Jaju and Amitabh Mehta for fruitful discussions.

6. REFERENCES

[1] E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *17th ICML*, 2000.

[2] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *The VLDB Journal*, 1998.

[3] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via ECOCs. *JAIR*, 2:263–286, 1995.

[4] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *7th CIKM*, 1998.

[5] Rayid Ghani. Using error-correcting codes for text classification. In *17th ICML*, 2000.

[6] Shantanu Godbole. Exploiting confusion matrices for automatic generation of topic hierarchies and scaling up multi-way classifiers. *Technical Report, IIT Bombay*, 2002. `http://www.it.iitb.ac.in/~shantanu/work/aps2002.pdf`

[7] Ryan Rifkin and Jason D. M. Rennie. Improving multi-class text classification with the support vector machine, AI Memo, AIM-2001-026, MIT, 2001.

[8] T. Joachims. A statistical learning model of text classification for SVMs. In *SIGIR 2001*, volume 24, ACM.

[9] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *14th ICML*, 1997.

[10] U. Kressel. Pairwise classification and support vector machines. In *Advances in Kernel Methods: Support Vector Learning*, MIT Press, 1999.

[11] T. M. Mitchell. Conditions for the equivalence of hierarchical and non-hierarchical Bayesian classifiers. Technical note, 1998. Online at `http://www.cs.cmu.edu/~tom/hierproof.ps`

[12] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In *Advances in NIPS 12*, MIT Press, 2000.

[13] M. F. Porter. An algorithm for suffix stripping. *Program, 14(3):130–137*, 1980.

[14] V. N. Vapnik. *The nature of statistical learning theory.* Springer Verlag, 1995.