Not Too Hot, Not Too Cold: The Bundled-SVM is Just Right!

Lawrence Shih KAI@AI.MIT.EDU
Yu-Han Chang YCHANG@AI.MIT.EDU
Jason Rennie JRENNIE@AI.MIT.EDU
David Karger KARGER@LCS.MIT.EDU
Artificial Intelligence Laboratory; Massachusetts Institute of Technology; Cambridge, MA 02139

Abstract

The Support Vector Machine (SVM) typically outperforms other algorithms on text classification problems, but requires training time roughly quadratic in the number of training documents. In contrast, linear time algorithms like Naive Bayes have lower performance, but can easily handle huge training sets. In this paper, we describe a technique that creates a continuum of classifiers between the SVM and a Naive Bayes like algorithm. Included in that continuum is a classifier that approximates SVM performance with linear training time. Another classifier on this continuum can outperform the SVM, yielding a breakeven point that beats other published results on Reuters-21578. We give empirical and theoretical evidence that our hybrid approach successfully navigates the tradeoffs between speed and performance.

1. Introduction & Related Work

There is a great need to find fast, effective algorithms for text classification. A KDD panel headed by Domingos [2002] discussed the tradeoff of speed and accuracy in the context of very large (e.g. 1 million record) databases. In particular, quadratic-time algorithms are not practical on such databases. This makes impractical the best-performing text classification algorithms. The Support Vector Machine (SVM) has consistently outperformed other algorithms [Yang and Liu, 1999; Joachims, 1997; Dumais *et al.*, 1998; Rennie and Rifkin, 2001] but requires approximately cn^2 time to train (Joachims estimates between $cn^{1.7}$ and $cn^{2.1}$), where n is the number of training examples, and c is an algorithm-dependent constant. K-nearest neighbors (kNN) performed nearly as well as the SVM in Yang and Liu's experiments, but labeling a document requires a full scan through the training set. Labeling a test set of size n requires cn^2 time. Linear Least Squares Fit (LLSF) performed well, but requires computation on all pairs of documents, giving it cn^2 running time [Yang and Liu, 1999]. Yang and Liu also experimented with a 3-layer Neural Network (NNet) with 64 hidden units, but it performed significantly worse than an SVM. Other tested algorithms, such as C4.5 and Bayes Nets, perform no better than the linear running time Rocchio and multinomial Naive Bayes (NB) algorithms. Rocchio and Naive Bayes are among the best linear-scaling algorithms and are practical for huge datasets. Since the best performing algorithms tend to be slowest, our focus is on finding algorithms that successfully navigate the tradeoffs between speed and performance.

We introduce a continuum of classifiers called the "Bundled-SVM" that spans the space between the high performance Support Vector Machine and a fast algorithm based on mean statistics that is similar to Naive Bayes. The Bundled-SVM uses the SVM as its core, but pre-processes the training data to offer varying degrees of efficiency. Given a set of documents, we randomly concatenate together documents within each class to produce a smaller set of documents that is handed to the SVM. The "bundle-size" parameter specifies how many documents are concatenated into each bundle. A Bundled-SVM with no concatenation is identical to the regular SVM. On the other hand, a Bundled-SVM with all documents concatenated together yields a classifier based on the mean statistics of each class, similar to Naive Bayes and Rocchio.

In this paper, we argue that the Bundled-SVM allows us to explicitly manage the trade-off between the accuracy of the SVM and the speed of a Naive Bayes like algorithm. We follow with experimental results that confirm these assertions. We show how the Bundled-SVM can be used as a linear time approximation to the SVM and give empirical evidence that this approximation generally outperforms existing linear-time algorithms. In one case, we find that the Bundled-SVM even outperforms the SVM, achieving the best published micro and macro breakeven points on the Reuters-21578 data set.

2. Basic Classification Algorithms

In this section, we describe three algorithms that are used in our experiments. The SVM and "Mean-frequency Discriminant" algorithms correspond to the two endpoints of our Bundled-SVM continuum. In our experiments, we also use multinomial NB as a baseline for comparison since it performs as well as (or better than) other common linear-time algorithms. Multinomial NB also serves as motivation for the Mean-frequency Discriminant algorithm.

Many of these algorithms use what is known as a document vector, d, which is a V-dimensional vector composed of word counts, where the value of the i^{th} component is the number of times word i occurred in the document, and V is the size of our total vocabulary. We apply the SMART ltc transform[1], which normalizes the document vectors to length one and yields improved performance [Yang and Liu, 1999].

2.1 The Support Vector Machine

The Support Vector Machine (SVM) is a classifier, originally proposed by Vapnik [1995], that finds a maximal margin separating hyperplane between two classes of data.

An SVM is trained via the following optimization problem:

$$\hat{w} = \mathrm{argmin}_w \frac{1}{2}\|w\|^2 + C\sum_i \xi_i, \tag{1}$$

with constraints

$$y_i(d_i \cdot w + b) \geq 1 - \xi_i \quad \forall i, \tag{2}$$

$$\xi_i \geq 0 \quad \forall i, \tag{3}$$

where each d_i is a document vector, y_i is the label ($+1$ or -1) for d_i and $\hat{w}$ is the vector of weights that defines the optimal separating hyperplane. This form of the optimization is called the "primal." By incorporating the inequality constraints via Lagrange multipliers, we arrive at the "dual" form of the problem,

$$\hat{w} = \mathrm{argmax}_w \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j (d_i \cdot d_j) \tag{4}$$

[1]The list of SMART weightings can be found at http://www.ai.mit.edu/∼jrennie/ecoc-svm/smart.html.

subject to

$$0 \leq \alpha_i \leq C \quad \forall i \tag{5}$$

$$\sum_i \alpha_i y_i = 0 \tag{6}$$

Given optimized values for the α_i, the optimal separating hyperplane is

$$\hat{w} = \sum_i \alpha_i y_i d_i. \tag{7}$$

For more information about the SVM, see Burges' tutorial [1998], Cristianini and Shawe-Taylor's book [2000], Evgeniou *et al.*'s article [2000] and Vapnik's book [1998].

The regular SVM finds a separating hyperplane defined by a weighted sum of the training documents. The Bundled-SVM, which concatenates together documents before giving them to the SVM, is similar to constraining some α's to be equal to each other. Section 3 provides a more complete description and theory of the Bundled-SVM.

2.2 Naive Bayes

Naive Bayes is a simple Bayesian text classification algorithm. We use the "multinomial" version that Yang and Liu [1999] used in their experiments. It assumes that each token in a document is drawn independently from a multinomial distribution; documents are classified according to the Bayes optimal decision rule. Each class, c, has its own vector of multinomial parameters, θ^c. θ^c_k is the (mean) rate of occurrence for word k in class c. Hence, the only statistics needed to learn a Naive Bayes model are the mean statistics. This is important for the comparison of Naive Bayes with the Bundled-SVM. The concatenation of documents retains the mean frequency statistics, so the one-bundle-per-class Bundled-SVM uses the same information as does Naive Bayes to learn a decision boundary.

We estimate parameters for Naive Bayes via Maximum a Posteriori (MAP) using the training data. A new document, d, is assigned the label $\hat{H}(d) = \mathrm{argmax}_c\, p(d|\hat{\theta}^c)p(c)$. Using D^c to denote the training data for class c, we use parameter estimates $\hat{\theta}^c = \mathrm{argmax}_\theta\, p(D^c|\theta)p(\theta)$. A Dirichlet parameter prior, $p(\theta)$, with hyper-parameters $\alpha_i = 2\ \forall i$ gives an estimate of $\hat{\theta}^c_k = \frac{N^c_k+1}{N^c+V}$ for word k, where N^c_k is the number of times word k occurred in class c, N^c is the total number of word occurrences in class c and V is the size of the vocabulary. The choice of the class prior has little effect on classification, so we choose a flat prior, $p(c) = \frac{1}{m}$. We assign a document, d, the label $\widehat{H}(d)$ where

$$\widehat{H}(d) = \mathrm{argmax}_c \prod_k \left(\frac{N_k^c + 1}{N^c + V} \right)^{f_k} \tag{8}$$

See Rennie [2001] for further explanation. Chakrabarti *et al.* [1997] use a flat parameter prior and expectation to derive the same parameter estimates. McCallum and Nigam [1998] give an explanation of the distinction between the traditional (independent features) Naive Bayes classifier and the multinomial.

2.3 Mean-frequency Discriminant

The fast end of the Bundled-SVM continuum concatenates together all of the documents in each class. It uses the same mean statistics as multinomial NB, but arrives at its decision boundary in a more direct way. For this reason, we call the algorithm the Mean-frequency Discriminant (MFD). Let d^c_{cat} be the document created by concatenating together all documents in class c. Let $d^c_{\text{cat}}(k)$ be the count of word k in d^c_{cat}. Then multinomial NB uses the statistics $N^c_k = d^c_{\text{cat}}(k)$ and $N^c = \sum_{k=1}^{V} d^c_{\text{cat}}(k)$. NB uses the assumption of a multinomial model to obtain a discriminant. MFD uses a different approach; it determines the decision boundary as an SVM does. For the binary problem, the discriminant is the perpendicular bisector of the vector from the negative class mean to the positive class mean. We do not use d^c_{cat} directly, because it is not normalized for length. We apply the SMART ltc transform to each d^c_{cat} and to test documents so that they are more easily comparable. Consider the binary classification problem with classes $c \in \{+, -\}$. Let d^c_{ltc} be the mean vector after the ltc transform has been applied. Then, classification of a document is equivalent to finding the mean vector, d^+_{ltc} or d^-_{ltc}, having the smallest angle with the test document. The MFD is equivalent to the binary classifier used by Bundled-SVM when all of the documents for each class are bundled together.

3. A New Approach

The Bundled-SVM, parametrized by the bundle size, generates a continuum of classifiers between the SVM and MFD. On the fast end of the continuum, MFD uses only the mean statistics of the data and thus achieves fast running time. MFD only requires one input point per class, namely the mean. Training and testing are clearly linear time. On the other end of the continuum, the SVM achieves high accuracy but requires a prohibitively long training time on large datasets. The SVM uses each individual document to train its classifier so it trains slower but has higher accuracy.

In order to achieve both high accuracy and fast training time, we explore the space of classifiers between MFD and the SVM. The Bundled-SVM is designed to combine the speed of MFD and the accuracy of the SVM by forming a new input data set that is smaller than the original data set but larger than the set of means used by MFD. Our approach proceeds as shown in Table 1. The bundle-size parameter, s, determines the size of our new input data set, and we will sometimes refer to the Bundled-SVM with parameter $s = k$ as a Bundled-SVM ($s = k$). The Bundled-SVM concatenates documents in the original data set together to form longer documents, each composed of s original documents of the same class.

The Bundled-SVM ($s = 1$) performs no concatenation and behaves identically to the SVM in both accuracy and training time. Training time is approximately cn^2, where n is the number of training examples, and accuracy is generally high. When each class is reduced to one point, denoted $s = \max$, the Bundled-SVM gives the SVM one point per class as input, namely the concatenation of all documents in a class. In the binary classification case, the SVM receives two vectors, one for each class, and the SVM will find the widest separating hyperplane between the two vectors, which happens to be the perpendicular bisector. This behavior is what we refer to as the Mean-frequency Discriminant algorithm. Training time becomes linear, cn, but accuracy is generally lower than in the $s = 1$ case.

In between these extreme values of the bundle-size $s = 1$ and $s = \max$, the training time is somewhere between cn and cn^2, and we expect the accuracy to be somewhere between the accuracies of the original SVM and MFD. It is clear that training time can be improved over the usual SVM as we decrease the size of the input data set by increasing the bundle size s. Accuracy improves over MFD as we decrease the bundle-size s because concatenating documents into bundles preserves more information about the distribution of the documents than the mean alone. Each bundle provides some additional information about the underlying structure of the distribution of document vectors within each class.

As a particularly noteworthy example of the tradeoff between accuracy and speed within this continuum, we can choose a bundle size of $s = \sqrt{n/m}$, where m is the number of classes given to us in the classification problem. Our approach proceeds by concatenating documents together on a per-class basis until we are left with $\sqrt{n/m}$ documents. We are thus left with at most $\sqrt{nm}$ documents. Using this set as our input data, the SVM training time is cn, since m, the number of classes, is fixed. As will be shown in Section 5, this Bundled-SVM ($s = \sqrt{n/m}$) performs nearly as well as the original SVM in terms of classification accuracy while achieving a dramatic speedup from quadratic to linear training time.

When the number of documents is not very large, a linear-time classifier is usually not necessary, but concatenating together documents can still be useful. For example, concatenating pairs of documents, thus halving the number of training documents, makes training time for the SVM a quarter of the time necessary to train on the full set of documents. Yet, we find that concatenating pairs of documents rarely hurts classification performance with the SVM and can even help.

3.1 Handling Large Data Sets

Often in text, however, the number of training documents is extremely large. A standard approach to handling exceedingly large amounts of data is to subsample. This makes state-of-the-art algorithms, such as the SVM, practical, but can greatly degrade performance. The ignored examples contain much information about the decision boundaries between classes. We propose that it is possible to retain some of this information by randomly concatenating together documents in the same class. Text is unusual in that it is very high dimensional. The set of possible decision boundaries between document classes is very large. As the SVM constructs the decision boundary as a weighted sum of training examples, the space of possible boundaries is often constrained by the number of training examples. We argue that concatenation is preferred to subsampling because the constraint it effectively imposes on the SVM is less severe than that of subsampling.

Let $D = \{d_1, \ldots, d_n\}$ be the set of training documents for a classification problem. We want to lessen the time it would take for an SVM to learn a decision boundary using these documents, so we consider two options, 1) subsampling, and 2) concatenation. For subsampling, we remove d_n from the set of training documents. Let $\alpha^s = \{\alpha_1^s, \ldots, \alpha_{n-1}^s\}$ be the set of α's that the SVM learns after we remove d_n. The decision boundary the SVM learns is $w^s = \sum_{i=1}^{n-1} \alpha_i^s y_i d_i$. For concatenation, we create a new set of training documents that contains one less document vector than contained in the original set, $D^c = \{d_1^c, \ldots, d_{n-1}^c\}$. The first $n-2$ documents are identical to the original training set, $d_i^c = d_i$ for $i \in \{1, \ldots, n-2\}$. d_{n-1}^c is defined as the document vector resulting from the concatenation of documents d_{n-1} and d_n (simply $d_{n-1} + d_n$). After training an SVM with this set of documents, D^c, we get a set of alphas, $\alpha^c = \{\alpha_1^c, \ldots, \alpha_{n-1}^c\}$, that define the decision boundary for the SVM, $w^c = \sum_{i=1}^{n-1} \alpha_i^c y_i d_i^c$.

Now, consider writing the two decision boundaries, w^s and w^c, in terms of α's on the full set of documents. w^s simply forces $\alpha_n = 0$. Let $\alpha_i := \alpha_i^s$ for $i \in \{1, \ldots, n-1\}$ and $\alpha_n := 0$. Then $w^s = \sum_{i=1}^{n} \alpha_i y_i d_i$. Subsampling is similar to training an SVM with the full set of documents but constraining some of the α's to be zero. Concatenation imposes a different constraint. Concatenating two documents corresponds to a summing of their document vectors. This forces two α's to be equal to each other. Let $\alpha_i := \alpha_i^c$ for $i \in \{1, \ldots, n-2\}$ and $\alpha_{n-1} := \alpha_n := \alpha_{n-1}^c$. Then, $w^c = \sum_{i=1}^{n} \alpha_i y_i d_i$. Concatenation is similar to training an SVM with the full set of documents, but constraining some α's to be equal to each other.

When the number of training documents (before concatenation or subsampling) is less than the size of the vocabulary ($n < V$), subsampling is clearly a poor choice—document vectors are usually linearly independent. We verified this on the three data sets we use for this paper. For each set of documents, only one or two can be removed without reducing the span of the set of document vectors. To see why this is expected, note that it is reasonable to assume document vectors are drawn from some smooth distribution. Consider drawing n points uniformly from a V-dimensional unit sphere. The set of points will be linearly independent with probability 1. The set of events where there is some linear dependence has measure zero. The same property is true of any smooth distribution in V-dimensional space. When all of the training examples given to an SVM are linearly independent, many α's will be non-zero. A large number of points will take part in defining the decision boundary. In this case, subsampling is a bad choice because it forces some of the α's to be zero. Concatenation is a better alternative; it allows all of the α's to be positive. The equality constraint means that the decision boundary may not be as good as the (unbundled) SVM, but it is a better approximation than the constraint that subsampling imposes.

When $n > V$, the argument for concatenation is less strong, but there is still reason to believe that it is a better alternative than subsampling. As with the bundled-SVM, to achieve linear time, roughly $\sqrt{n}$ documents need to be subsampled; and in most cases (when $n < V^2$) this lowers n to below V. Even in this case, where the document vectors are no longer linearly independent, the decision boundary is not defined by an arbitrary linear combination of document vectors, but rather by a non-negative combination. Removing a vector may reduce the expressiveness of the set. In other words, when a document is removed by subsampling, the set of feasible decision boundaries may be reduced even though that document is a weighted sum of other documents. Concatenation also does not fully preserve the set of possible decision boundaries, but it does preserve some of the information from each data point, whereas subsampling completely eliminates a set of documents that may be useful in determining the decision boundary.

3.2 Further Analysis

Higher levels of bundling lessens the amount of information available to the SVM. However, there is reason to believe that the Bundled-SVM still has most of the information it needs to do classification. The document vectors in text are extremely sparse; of the entire vocabulary (typically in the tens of thousands), only a hundred words are normally present in any given document. It is fairly common for words in the training set to be exclusively in one class or the other. For example, if we create twenty one-vs-all classifiers on 20 Newsgroups with 2000 training examples, we find that the average classifier has 808 words that exclusively sit in the "one" class, and 24936 words that are exclusively in the "all" class. Only 3167 words appear in both classes.

When the SVM (or most other classifiers) creates a decision boundary for such data, each of the 808 words in the "one" class will receive positive weight and each of the 24936 words in the "all" class will receive negative weight. Concatenating the documents has no effect on these statistics; the SVM will maintain a positive or negative weight on the exclusive words irrespective of the bundle size parameter. The separating hyperplane may rotate as the bundle size changes, but since about 90% of the words are exclusive to the "one" or the "all" class, the vast majority of weights will maintain their sign.

4. Data Sets and Experimental Setup

For our experiments, we use three well-known data sets: 20 Newsgroups [McCallum and Nigam, 1998; Slonim and Tishby, 1999; Berger, 1999], Industry Sector [Ghani, 2000] and Reuters-21578. [Yang and Liu, 1999; Joachims, 1997; Schapire and Singer, 2000]. We use Rainbow to pre-process the raw documents into feature vectors [McCallum, 1996]; our pre-processing steps mimic those used by Rennie and Rifkin for 20 Newsgroups and Industry Sector; we use the ModApte split for Reuters-21578.

Let s be the desired bundle-size.
Let n_i be the number of documents in class i.
Let $g_{ij} = 1$ for each document d_{ij} in class i.
Let $D_i = \{d_{i1}, \ldots, d_{in_i}\}$.
for $j = 1$ to n_i/s

- Randomly select two document indices, a and b, such that $g_a + g_b < s$.
- Remove documents d_a and d_b from D_i.
- Concatenate d_a and d_b and insert this new document as d_{n_i+j} into D_i. Set $g_{n_i+j} = g_a + g_b$.

Table 1. Subroutine used by the Bundled-SVM to concatenate documents together. To train the Bundled-SVM, the resulting D_i is then used as the new, smaller input data set for a regular SVM.

The 20 Newsgroups data set is a collection of Usenet posts, organized by newsgroup category, which was first collected as a text corpus by Lang [1995][2]. It contains 19,997 documents evenly distributed across 20 classes. We remove all headers and UU-encoded blocks, and skip stoplist words and words that occur only once[3]. The vocabulary size is 62,061.

The Industry Sector data is a collection of corporate web pages organized into categories based on what a company produces or does[4]. There are 9649 documents and 105 categories. The largest category has 102 documents, the smallest has 27. We remove headers, and skip stoplist words and words that occur only once[5]. The vocabulary size is 55,197.

Since 20 Newsgroups and Industry Sector are multiclass, single-label problems, we constructed a one-vs-all classifier for each category. To assign a label to a document, we selected the most confident classifier. Note we bundled only within the top-level categories. For example, on the 20 Newsgroups data set, the Bundled-SVM (s = max) used 1 positive and 19 negative training examples for each classifier.

The Reuters-21578 is a collection of Reuters newswire stories that is commonly used in text classification ex-

[2] The 20 Newsgroups data set can be obtained from http://www.ai.mit.edu/~jrennie/20Newsgroups/.

[3] Our 20 Newsgroups pre-processing corresponds to rainbow options "–istext-avoid-uuencode –skip-header -O 2."

[4] We obtained the Industry Sector data set from http://www-2.cs.cmu.edu/~TextLearning/datasets.html.

[5] Our Industry Sector pre-processing corresponds to rainbow options "–skip-header -O 2."

periments. We use the Mod-Apte split. There are 90 categories with at least one document in both the training and the test set. After eliminating documents not labeled with at least one of these categories, we are left with 7770 training documents and 3019 test documents. After eliminating words from a standard stop list and words that only appear once, we have a vocabularly size of 18,624. Reuters poses a multi-label problem. We construct a one-vs-all classifier for each category. A document is assigned a label for each classifier that produces a positive value.

For 20 Newsgroups and Industry Sector, we use multi-class classification accuracy to compare different algorithms. For Reuters, we use precision-recall breakeven. To compute breakeven, for each class, we trade-off between precision and recall until we find their scores to be equal. For macro breakeven, we average these scores. For micro breakeven, we perform a weighted average, where the weight for a class is the number of testing examples in that class.

The Mod-Apte split defines the training and test sets to be used for Reuters. For the 20 Newsgroups and Industry Sector, we experiment on 10 random test/train splits and report average performance. For each split, we select the training set first and use all remaining documents for testing. For the 20 Newsgroups data, we created training sets with 600, 300, 150, 100, 50, 25, and 10 training documents per class; and for the Industry Sector data, we used training sets of (up to) 52, 20, 10, and 5 training documents per class.

We compare four different algorithms on the 20 Newsgroup and Industry Sector datasets: a standard SVM classifier, a multinomial NB classifier, the Bundled-SVM classifier, and a subsampled SVM classifier. The subsampled SVM uses a number of training documents equal to the number of bundles in the Bundled-SVM. In particular, it chooses the longest documents—we found that this worked better than other subsampling techniques. The SVM and NB are described in section 2, and were chosen as high-accuracy algorithms relative to their respective training times of cn^2 and cn.

For our SVM and Bundled-SVM experiments, we use the SMART ltc transform; the SvmFu package is used for running experiments [Rifkin, 2000]. We set C, the penalty for misclassifying training points, at 10. We produce multi-class labels by training a one-versus-all SVM for each class and assigning the label of the most confident SVM. We use the linear kernel for the SVM since after applying the SMART ltc transform, the linear kernel performs as well as non-linear kernels in text classification [Yang and Liu, 1999].

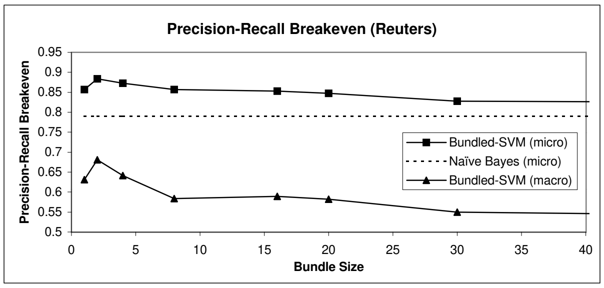

Figure 1. The performance of various Bundled-SVM classifiers on the Reuters dataset is shown. At $s = 2$, the micro-averaged precision-recall score of .884 exceeds previously published results. For bundle sizes between 2 and 9, Bundled-SVM performs at least as well as a regular SVM. The Bundled-SVM always performs significantly better than Naive Bayes

5. Results

In Section 3, we gave reasons for why the Bundled-SVM should outperform a subsampled SVM and will roughly approximate a normal SVM. Our experimental evidence over three datasets shows that this approximation is very good indeed. Moreover, the training time is reduced significantly, but the performance is still superior to subsampling and other linear time algorithms such as Naive Bayes.

5.1 The Reuters-21578 Dataset

Figure 1 shows that the Bundled-SVM performs very well on the Reuters dataset. When measured with the micro-averaged precision-recall metric, the Bundled-SVM ($s = 2$) sets a record high score of .884, three percentage points better than the SVM on the same dataset. Note that our SVM score (.857) is close to what Yang and Liu [1999] reports. This beats the score of .878 published by Weiss *et al.* [1999]. At the point the Bundled-SVM scales linearly ($s = \sqrt{n/m} = \sqrt{7770/90} = 9$), it achieves approximately the same breakeven as the regular SVM of .857 in significantly less time.

The results are mostly consistent with our expectations; as the bundle-size increases, after $s = 2$, the performance gradually decreases, mirroring our expected transition between the SVM ($s = 1$) and the fully bundled algorithm which uses $s = \max$. This shows that the Bundled-SVM successfully navigates the tradeoffs between speed and performance.

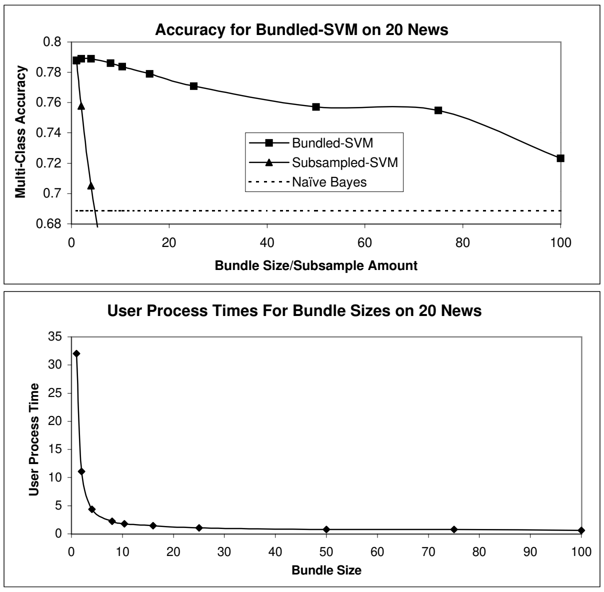

Figure 2. The performance of Bundled-SVM with different bundle sizes is shown (top) for the 20 News dataset with 2000 training examples. The running time decreases exponentially (bottom), but the accuracy only gradually decreases and always remains above Naive Bayes (the dotted, horizontal line).

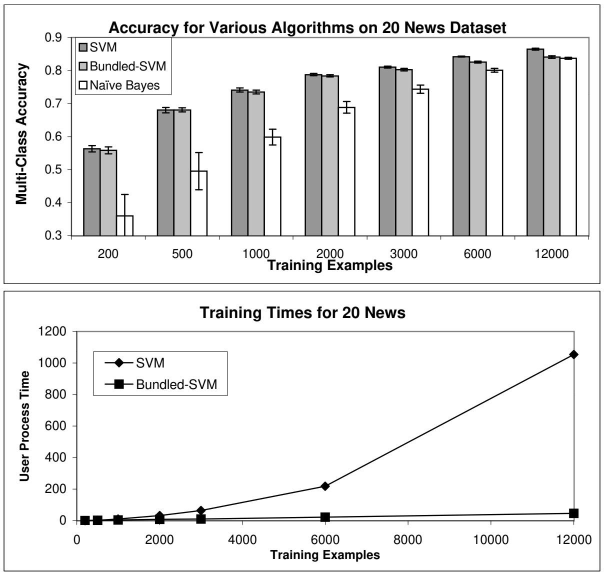

Figure 3. Bundled-SVM ($s = \sqrt{n/m}$) performs between the SVM and Naive Bayes (top). Its training time scales linearly (bottom) while the SVM scales quadratically. Thus the Bundled-SVM achieves a performance improvement over Naive Bayes while training much more quickly than the SVM.

5.2 The 20 News Dataset

The 20 News dataset has some notable differences with Reuters: it has far fewer classes, more total documents, and each document has exactly one label. Because its document size is the largest of the three datasets, we used 20 News to show how classification time changes with both the number of training documents and with the bundle size. Our first pair of experiments, displayed in Figure 2, shows how the bundle size effects performance and user process time on the dataset with 2,000 training examples. The performance with a bundle size of 2 is approximately even with no bundling, followed by a decrease in performance as the bundle size increases. The decrease in performance is more substantial than on Reuters, but all the bundle sizes from no bundling to full bundling (100 points/class) perform better than Naive Bayes.

The measured user process time decreases exponentially with s, the bundle size. An s-sized bundle causes a reduction in the number of input documents for class i from n_i to $\min(n_i/s, 1)$, which changes the speed of the algorithm from roughly cn^2 to roughly $c(n/s)^2$. At the linear time point, ($s = \sqrt{n/m} = \sqrt{2000/20} = 10$), the performance is only slightly worse than the SVM, but the Bundled-SVM takes a very small fraction of the time that the SVM requires. This linear time point demonstrates the Bundled-SVM's successful trade-off between performance and speed: the region where the performance drop-off is most gradual corresponds to the area where the absolute time speedup is greatest.

Our second pair of experiments fixes the bundle-size at $\sqrt{n/m}$ and varies the number of training examples (Figure 3). At that bundle-size, Bundled-SVM scales linearly with the number of training examples (bottom graph) but maintains a high multi-class accuracy within 2% of the regular SVM. It consistently outperforms Naive Bayes over several training sizes. The difference between Bundled-SVM and Naive Bayes is largest for small training sizes, suggesting Bundled-SVM does a good job generalizing over small data sets.

5.3 The Industry Sector Dataset

The Industry Sector database provides a third empirical validation of our method (Figure 4). The top graph shows Bundled-SVM outperforms Naive Bayes. However, unlike the first two datasets, performance does not gradually descend as bundle-size increases. Rather, it is constrained to a fairly small range. The regular SVM ($s = 1$) does not perform much better than the SVM with one bundle per class ($s = \max$). The bottom graph shows the performance of the linear time Bundled-SVM ($s = \sqrt{n/m}$) performs somewhat

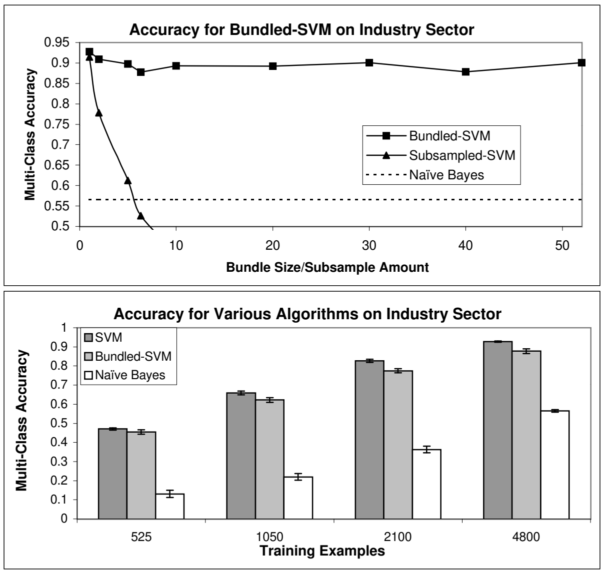

Figure 4. Industry Sector performance across various bundle-sizes (top) and training sizes (bottom). The top graph shows that the Bundled-SVM's performance does not change much after an initial drop. There is little performance difference between $s = 10$ and $s = \max$. But, at all bundle-sizes, it significantly outperforms Naive Bayes (dotted horizontal line). The bottom graph shows the performance of the linear-time Bundled-SVM ($s = \sqrt{n/m}$). Its performance closely approximates the SVM and outperforms Naive Bayes.

worse than the SVM, but that both perform far better than Naive Bayes.

5.4 Summary of Results

Table 2 summarizes our results for four classifiers on the Bundled-SVM continuum. The results are for 20 Newsgroups with 2000 training points, Industry Sector with 4800 training points, and the Reuters Mod-Apte split with 7770 training points. The scores are in terms of accuracy for 20 News and Industry Sector, and precision-recall breakeven for Reuters.

6. Conclusions and Future Work

We presented the Bundled-SVM, which creates a continuum of classifiers between the SVM and the MFD algorithm. We argued that this continuum explicitly allows us to trade-off between the performance of the SVM and the fast time of algorithms that use mean statistics, like Naive Bayes and Rocchio. We also showed that our technique of modifying the training data by concatenating documents is superior to subsampling.

Along the Bundled-SVM continuum, two points are worth highlighting that empirically give good results. Bundled-SVM ($s = 2$), which concatenates pairs of documents by class, runs roughly four times faster than the SVM; and yet, in two of three datasets, it outperforms the SVM (in the case of Reuters, significantly). The Bundled-SVM ($s = \sqrt{n/m}$) runs in linear time but outperforms Naive Bayes on all three datasets.

When using one bundle per class, the Bundled-SVM uses information similar to Naive Bayes and Rocchio in constructing a classifier. But, the classification boundary is different. Since Naive Bayes and Rocchio are popular and effective algorithms when there is plentiful training data, we plan to create versions of the Bundled-SVM that exactly match these algorithms when using one bundle per class. This would give a guaranteed level of performance at one end of the continuum and the potential for much improved classification at smaller bundle-sizes.

An important next step for this work is to apply the Bundled-SVM to much larger data sets. We have shown the ability to trade-off classification performance for speed on moderate-size data sets, but linear-time algorithms only become essential on larger corpuses. An evaluation on a much larger data set will make clear whether the performance observed so far holds as the number of training examples increases.

Acknowledgements This work was supported in part by the MIT Oxygen Partnership and Graduate Research Fellowships from the National Science Foundation. We thank Nati Srebro and Poompat Saengudomlert for comments on this paper.

References

[Berger, 1999] Adam Berger. Error-correcting output coding for text classification. In *Proceedings of IJCAI-99 Workshop on Machine Learning for Information Filtering*, Stockholm, Sweeden, 1999.

[Burges, 1998] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[Chakrabarti *et al.*, 1997] Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. Using taxonomy, discriminants and signatures for navigating in text databases. In *Proceedings of the 23rd VLDB Conference*, 1997.

[Cristianini and Shawe-Taylor, 2000] Nello Cristianini and John Shawe-Taylor. *An introduction to sup-*

Dataset	$s = 1$ (SVM)	$s = 2$	$s = \sqrt{n/m}$	$s = \max$	NB
Reuters (micro)	0.857	0.884	0.857	0.708	0.736
Reuters (macro)	0.631	0.681	0.586	0.508	0.264
20 News	0.788	0.788	0.784	0.723	0.689
Industry Sector	0.928	0.909	0.878	0.901	0.566

Table 2. Four classifiers along the Bundled-SVM continuum are compared to Naive Bayes. The Bundle-SVMs correspond to $s = 1$ (SVM), $s = 2$, $s = \sqrt{n/m}$ (linear-time), and $s = \max$ (MFD). Results are shown for three datasets. Results for Reuters are given for both micro and macro breakeven; other numbers are multi-class accuracy.

port vector machines. Cambridge University Press, 2000.

[Domingos, 2002] Pedro Domingos. When and how to subsample: Report on the kdd-2001 panel. *SIGKDD Explorations*, 3(2), 2002.

[Dumais *et al.*, 1998] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and reepresentations for text classification. In *Seventh International Conference on Information and Knowledge Management*, 1998.

[Evgeniou *et al.*, 2000] Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

[Ghani, 2000] Rayid Ghani. Using error-correcting codes for text classification. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[Joachims, 1997] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

[Joachims, 1999] Thorsten Joachims. Making large-scale svm learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[Lang, 1995] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[McCallum and Nigam, 1998] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *Proceedings of the AAAI-98 workshop on Learning for Text Categorization*, 1998.

[McCallum, 1996] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/∼mccallum/bow, 1996.

[Rennie and Rifkin, 2001] Jason D. M. Rennie and Ryan Rifkin. Improving multiclass text classification with the Support Vector Machine. Technical Report AIM-2001-026, Massachusetts Insititute of Technology, Artificial Intelligence Laboratory, 2001.

[Rennie, 2001] Jason D. M. Rennie. Improving multi-class text classification with naive bayes. Master's thesis, Massachusetts Institute of Technology, 2001.

[Rifkin, 2000] Ryan Rifkin. Svmfu. http://five-percent-nation.mit.edu/SvmFu/, 2000.

[Schapire and Singer, 2000] Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, pages 135–168, 2000.

[Slonim and Tishby, 1999] Noam Slonim and Naftali Tishby. Agglomerative information bottleneck. In *Neural Information Processing Systems 12 (NIPS-99)*, 1999.

[Vapnik, 1995] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[Vapnik, 1998] Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[Weiss *et al.*, 1999] S. M. Weiss, C. Apte, F. J. Damerau, D. E. Johnson, J. F. Oles, T. Goetz, and T. Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14(4):63–69, 1999.

[Yang and Liu, 1999] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.