Relaxed Online SVMs for Spam Filtering

D. Sculley
Tufts University
Department of Computer Science
161 College Ave., Medford, MA USA
dsculleycs.tufts.edu

Gabriel M. Wachman
Tufts University
Department of Computer Science
161 College Ave., Medford, MA USA
gwachm01cs.tufts.edu

ABSTRACT

Spam is a key problem in electronic communication, including large-scale email systems and the growing number of blogs. Content-based filtering is one reliable method of combating this threat in its various forms, but some academic researchers and industrial practitioners disagree on how best to filter spam. The former have advocated the use of Support Vector Machines (SVMs) for content-based filtering, as this machine learning methodology gives state-of-the-art performance for text classification. However, similar performance gains have yet to be demonstrated for online spam filtering. Additionally, practitioners cite the high cost of SVMs as reason to prefer faster (if less statistically robust) Bayesian methods. In this paper, we offer a resolution to this controversy. First, we show that online SVMs indeed give state-of-the-art classification performance on online spam filtering on large benchmark data sets. Second, we show that nearly equivalent performance may be achieved by a Relaxed Online SVM (ROSVM) at greatly reduced computational cost. Our results are experimentally verified on email spam, blog spam, and splog detection tasks.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – spam

General Terms

Measurement, Experimentation, Algorithms

Keywords

support vector machines, spam filtering, blogs, splogs

1. INTRODUCTION

Electronic communication is increasingly plagued by unwanted or harmful content known as *spam*. The most well known form of spam is *email spam*, which remains a major problem for large email systems. Other forms of spam are also becoming problematic, including *blog spam*, in which spammers post unwanted comments in blogs [21], and *splogs*, which are fake blogs constructed to enable *link spam* with the hope of boosting the measured importance of a given webpage in the eyes of automated search engines [17]. There are a variety of methods for identifying these many forms of spam, including compiling blacklists of known spammers, and conducting link analysis.

The approach of *content analysis* has shown particular promise and generality for combating spam. In content analysis, the actual message text (often including hyper-text and meta-text, such as HTML and headers) is analyzed using machine learning techniques for text classification to determine if the given content is spam. Content analysis has been widely applied in detecting email spam [11], and has also been used for identifying blog spam [21] and splogs [17]. In this paper, we do not explore the related problem of *link spam*, which is currently best combated by link analysis [13].

1.1 An Anti-Spam Controversy

The anti-spam community has been divided on the choice of the best machine learning method for content-based spam detection. Academic researchers have tended to favor the use of Support Vector Machines (SVMs), a statistically robust machine learning method [7] which yields state-of-the-art performance on general text classification [14]. However, SVMs typically require training time that is quadratic in the number of training examples, and are impractical for large-scale email systems. Practitioners requiring content-based spam filtering have typically chosen to use the faster (if less statistically robust) machine learning method of Naive Bayes text classification [11, 12, 20]. This Bayesian method requires only linear training time, and is easily implemented in an online setting with incremental updates. This allows a deployed system to easily adapt to a changing environment over time. Other fast methods for spam filtering include compression models [1] and logistic regression [10]. It has not yet been empirically demonstrated that SVMs give improved performance over these methods in an online spam detection setting [4].

1.2 Contributions

In this paper, we address the anti-spam controversy and offer a potential resolution. We first demonstrate that online SVMs do indeed provide state-of-the-art spam detection through empirical tests on several large benchmark data sets of email spam. We then analyze the effect of the *tradeoff*

SIGIR'07, July 23–27, 2007, Amsterdam, The Netherlands.

parameter in the SVM objective function, which shows that the expensive SVM methodology may, in fact, be overkill for spam detection. We reduce the computational cost of SVM learning by relaxing this requirement on the maximum margin in online settings, and create a Relaxed Online SVM, ROSVM, appropriate for high performance content-based spam filtering in large-scale settings.

2. SPAM AND ONLINE SVMS

The controversy between academics and practitioners in spam filtering centers on the use of SVMs. The former advocate their use, but have yet to demonstrate strong performance with SVMs on online spam filtering. Indeed, the results of [4] show that, when used with default parameters, SVMs actually perform *worse* than other methods. In this section, we review the basic workings of SVMs and describe a simple Online SVM algorithm. We then show that Online SVMs indeed achieve state-of-the-art performance on filtering email spam, blog comment spam, and splogs, so long as the tradeoff parameter C is set to a high value. However, the cost of Online SVMs turns out to be prohibitive for large-scale applications. These findings motivate our proposal of Relaxed Online SVMs in the following section.

2.1 Background: SVMs

SVMs are a robust machine learning methodology which has been shown to yield state-of-the-art performance on text classification [14]. by finding a hyperplane that separates two classes of data in data space while maximizing the margin between them.

We use the following notation to describe SVMs, which draws from [23]. A data set X contains n labeled example vectors $\{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_n, y_n)\}$, where each $\mathbf{x}_i$ is a vector containing features describing example i, and each y_i is the class label for that example. In spam detection, the classes spam and ham (*i.e.*, not spam) are assigned the numerical class labels $+1$ and -1, respectively. The linear SVMs we employ in this paper use a *hypothesis* vector $\mathbf{w}$ and *bias* term b to classify a new example $\mathbf{x}$, by generating a predicted class label $f(x)$:

$$f(x) = sign(< \mathbf{w}, \mathbf{x} > +b)$$

SVMs find the hypothesis $\mathbf{w}$, which defines the separating hyperplane, by minimizing the following objective function over all n training examples:

$$\tau(\mathbf{w}, \xi) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=i}^{n} \xi_i$$

under the constraints that

$$\forall i = \{1..n\} : y_i(< \mathbf{w}, \mathbf{x}_i > +b) \geq 1 - \xi_i, \xi_i \geq 0$$

In this objective function, each slack variable ξ_i shows the amount of error that the classifier makes on a given example $\mathbf{x}_i$. Minimizing the sum of the slack variables corresponds to minimizing the loss function on the training data, while minimizing the term $\frac{1}{2}||\mathbf{w}||^2$ corresponds to maximizing the margin between the two classes [23]. These two optimization goals are often in conflict; the tradeoff parameter C determines how much importance to give each of these tasks.

Linear SVMs exploit data sparsity to classify a new instance in $O(s)$ time, where s is the number of non-zero features. This is the same classification time as other linear classifiers, and as Naive Bayesian classification. Training SVMs, however, typically takes $O(n^2)$ time, for n training examples. A variant for linear SVMs was recently proposed which trains in $O(ns)$ time [15], but because this method has a high constant, we do not explore it here.

```
GIVEN: DATA SET X = (x_1, y_1), ..., (x_n, y_n), C, m:

INITIALIZE w := 0, b := 0, SEENDATA := { }
FOR EACH x_i ∈ X DO:
        CLASSIFY x_i USING f(x_i) = sign(< w, x_i > +b)
        IF y_i f(x_i) < 1
                FIND w', b' USING SMO ON SEENDATA,
                USING w, b AS SEED HYPOTHESIS.
        ADD x_i TO SEENDATA
DONE
```

Figure 1: Pseudo code for Online SVM.

2.2 Online SVMs

In many traditional machine learning applications, SVMs are applied in *batch mode*. That is, an SVM is trained on an entire set of training data, and is then tested on a separate set of testing data. Spam filtering is typically tested and deployed in an *online* setting, which proceeds incrementally. Here, the learner classifies a new example, is told if its prediction is correct, updates its hypothesis accordingly, and then awaits a new example. Online learning allows a deployed system to adapt itself in a changing environment.

Re-training an SVM from scratch on the entire set of previously seen data for each new example is cost prohibitive. However, using an old hypothesis as the starting point for re-training reduces this cost considerably. One method of incremental and decremental SVM learning was proposed in [2]. Because we are only concerned with incremental learning, we apply a simpler algorithm for converting a batch SVM learner into an online SVM (see Figure 1 for pseudo-code), which is similar to the approach of [16].

Each time the Online SVM encounters an example that was poorly classified, it retrains using the old hypothesis as a starting point. Note that due to the Karush-Kuhn-Tucker (KKT) conditions, it is not necessary to re-train on well-classified examples that are outside the margins [23].

We used Platt's SMO algorithm [22] as a core SVM solver, because it is an iterative method that is well suited to converge quickly from a good initial hypothesis. Because previous work (and our own initial testing) indicates that binary feature values give the best results for spam filtering [20, 9], we optimized our implementation of the Online SMO to exploit fast inner-products with binary vectors. [1]

2.3 Feature Mapping Spam Content

Extracting machine learning features from text may be done in a variety of ways, especially when that text may include hyper-content and meta-content such as HTML and header information. However, previous research has shown that simple methods from text classification, such as bag of words vectors, and overlapping character-level n-grams, can achieve strong results [9]. Formally, a bag of words vector is a vector $\mathbf{x}$ with a unique dimension for each possible

[1]Our source code is freely available at `www.cs.tufts.edu/~dsculley/onlineSMO`.

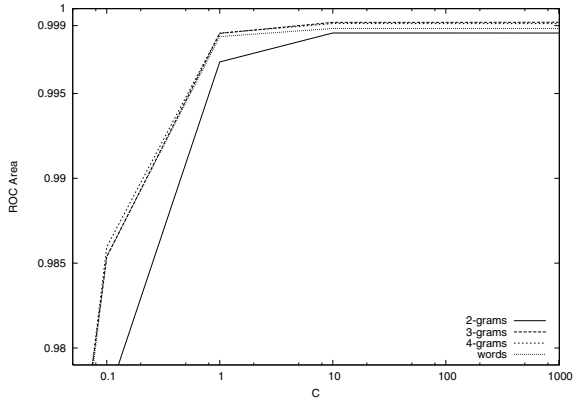

Figure 2: Tuning the Tradeoff Parameter C. Tests were conducted with Online SMO, using binary feature vectors, on the spamassassin data set of 6034 examples. Graph plots C versus Area under the ROC curve.

word, defined as a contiguous substring of non-whitespace characters. An n-gram vector is a vector $\mathbf{x}$ with a unique dimension for each possible substring of n total characters. Note that n-grams may include whitespace, and are overlapping. We use binary feature scoring, which has been shown to be most effective for a variety of spam detection methods [20, 9]. We normalize the vectors with the Euclidean norm. Furthermore, with email data, we reduce the impact of long messages (for example, with attachments) by considering only the first 3,000 characters of each string. For blog comments and splogs, we consider the whole text, including any meta-data such as HTML tags, as given. No other feature selection or domain knowledge was used.

2.4 Tuning the Tradeoff Parameter, C

The SVM tradeoff parameter C must be tuned to balance the (potentially conflicting) goals of maximizing the margin and minimizing the training error. Early work on SVM based spam detection [9] showed that high values of C give best performance with binary features. Later work has not always followed this lead: a (low) default setting of C was used on splog detection [17], and also on email spam [4].

Following standard machine learning practice, we tuned C on separate tuning data not used for later testing. We used the publicly available `spamassassin` email spam data set, and created an online learning task by randomly interleaving all 6034 labeled messages to create a single ordered set.

For tuning, we performed a coarse parameter search for C using powers of ten from .0001 to 10000. We used the Online SVM described above, and tested both binary bag of words vectors and n-gram vectors with $n = \{2, 3, 4\}$. We used the first 3000 characters of each message, which included header information, body of the email, and possibly attachments. Following the recommendation of [6], we use Area under the ROC curve as our evaluation measure. The results (see Figure 2) agree with [9]: there is a plateau of high performance achieved with all values of $C \geq 10$, and performance degrades sharply with $C < 1$. For the remainder of our experiments with SVMs in this paper, we set $C = 100$. We will return to the observation that very high values of C do not degrade performance as support for the intuition that relaxed SVMs should perform well on spam.

Table 1: Results for Email Spam filtering with Online SVM on benchmark data sets. Score reported is (1-ROCA)%, where 0 is optimal.

	TREC05P-1	TREC06P
ONSVM: WORDS	0.015 (.011-.022)	0.034 (.025-.046)
3-GRAMS	0.011 (.009-.015)	0.025 (.017-.035)
4-GRAMS	0.008 (.007-.011)	0.023 (.017-.032)
SPAMPROBE	0.059 (.049-.071)	0.092 (.078-.110)
BOGOFILTER	0.048 (.038-.062)	0.077 (.056-.105)
TREC WINNERS	0.019 (.015-.023)	0.054 (.034-.085)
53-ENSEMBLE	0.007 (.005-.008)	0.020 (.007-.050)

Table 2: Results for Blog Comment Spam Detection using SVMs and Leave One Out Cross Validation. We report the same performance measures as in the prior work for meaningful comparison.

	ACCURACY	PRECISION	RECALL
SVM $C = 100$: WORDS	0.931	0.946	0.954
3-GRAMS	0.951	0.963	0.965
4-GRAMS	0.949	0.967	0.956
PRIOR BEST METHOD	0.83	0.874	0.874

2.5 Email Spam and Online SVMs

With C tuned on a separate tuning set, we then tested the performance of Online SVMs in spam detection. We used two large benchmark data sets of email spam as our test corpora. These data sets are the 2005 TREC public data set `trec05p-1` of 92,189 messages, and the 2006 TREC public data sets, `trec06p`, containing 37,822 messages in English. (We do not report our strong results on the `trec06c` corpus of Chinese messages as there have been questions raised over the validity of this test set.) We used the canonical ordering provided with each of these data sets for fair comparison.

Results for these experiments, with bag of words vectors and and n-gram vectors appear in Table 1. To compare our results with previous scores on these data sets, we use the same (1-ROCA)% measure described in [6], which is one minus the area under the ROC curve, expressed as a percent. This measure shows the percent chance of error made by a classifier asserting that one message is more likely to be spam than another. These results show that Online SVMs do give state of the art performance on email spam. The only known system that out-performs the Online SVMs on the `trec05p-1` data set is a recent ensemble classifier which combines the results of 53 unique spam filters [19]. To our knowledge, the Online SVM has out-performed every other single filter on these data sets, including those using Bayesian methods [5, 3], compression models [5, 3], logistic regression [10], and perceptron variants [3], the TREC competition winners [5, 3], and open source email spam filters `BogoFilter v1.1.5` and `SpamProbe v1.4d`.

2.6 Blog Comment Spam and SVMs

Blog comment spam is similar to email spam in many regards, and content-based methods have been proposed for detecting these spam comments [21]. However, large benchmark data sets of labeled blog comment spam do not yet exist. Thus, we run experiments on the only publicly available data set we know of, which was used in content-based blog

Table 3: Results for Splog vs. Blog Detection using SVMs and Leave One Out Cross Validation. We report the same evaluation measures as in the prior work for meaningful comparison.

FEATURES	PRECISION	RECALL	F1
SVM $C = 100$: WORDS	0.921	0.870	0.895
3-GRAMS	0.904	0.866	0.885
4-GRAMS	0.928	0.876	**0.901**
PRIOR SVM WITH: WORDS	0.887	0.864	0.875
4-GRAMS	0.867	0.844	0.855
WORDS+URLS	0.893	0.869	0.881

comment spam detection experiments by [21]. Because of the small size of the data set, and because prior researchers did not conduct their experiments in an on-line setting, we test the performance of linear SVMs using leave-one-out cross validation, with SVM-Light, a standard open-source SVM implementation [14]. We use the parameter setting $C = 100$, with the same feature space mappings as above. We report accuracy, precision, and recall to compare these to the results given on the same data set by [21]. These results (see Table 2) show that SVMs give superior performance on this data set to the prior methodology.

2.7 Splogs and SVMs

As with blog comment spam, there is not yet a large, publicly available benchmark corpus of labeled splog detection test data. However, the authors of [17] kindly provided us with the labeled data set of 1,389 blogs and splogs that they used to test content-based splog detection using SVMs. The only difference between our methodology and that of [17] is that they used default parameters for C, which SVM-Light sets to $\frac{1}{avg||\mathbf{x}||_2}$. (For normalized vectors, this default value sets $C = 1$.) They also tested several domain-informed feature mappings, such as giving special features to url tags.

For our experiments, we used the same feature mappings as above, and tested the effect of setting $C = 100$. As with the methodology of [17], we performed leave one out cross validation for apples-to-apples comparison on this data. The results (see Table 3) show that a high value of C produces higher performance for the same feature space mappings, and even enables the simple 4-gram mapping to out-perform the previous best mapping which incorporated domain knowledge by using words and urls.

2.8 Computational Cost

The results presented in this section demonstrate that linear SVMs give state of the art performance on content-based spam filtering. However, this performance comes at a price. Although the blog comment spam and splog data sets are too small for the quadratic training time of SVMs to appear problematic, the email data sets are large enough to illustrate the problems of quadratic training cost.

Table 4 shows computation time versus data set size for each of the online learning tasks (on same system). The training cost of SVMs are prohibitive for large-scale content based spam detection, or a large blog host. In the following section, we reduce this cost by relaxing the expensive requirements of SVMs.

FEATURES	TREC06P	TREC05P-1
WORDS	12196S	66478S
3-GRAMS	44605S	128924S
4-GRAMS	87519S	242160S
CORPUS SIZE	32822	92189

Table 4: Execution time for Online SVMs with email spam detection, in CPU seconds. These times do not include the time spent mapping strings to feature vectors. The number of examples in each data set is given in the last row as corpus size.

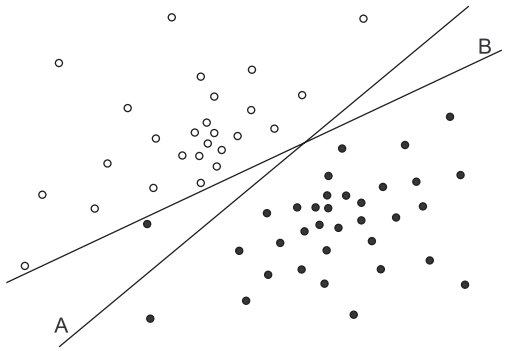

Figure 3: Visualizing the effect of C. Hyperplane A maximizes the margin while accepting a small amount of training error. This corresponds to setting C to a low value. Hyperplane B accepts a smaller margin in order to reduce training error. This corresponds to setting C to a high value. Content-based spam filtering appears to do best with high values of C.

3. RELAXED ONLINE SVMS (ROSVM)

One of the main benefits of SVMs is that they find a decision hyperplane that maximizes the margin between classes in the data space. Maximizing the margin is expensive, typically requiring quadratic training time in the number of training examples. However, as we saw in the previous section, the task of content-based spam detection is best achieved by SVMs with a high value of C. Setting C to a high value for this domain implies that minimizing training loss is more important than maximizing the margin (see Figure 3).

Thus, while SVMs do create high performance spam filters, applying them in practice is overkill. The full margin maximization feature that they provide is unnecessary, and relaxing this requirement can reduce computational cost. We propose three ways to relax Online SVMs:

- Reduce the size of the optimization problem by only optimizing over the last p examples.
- Reduce the number of training updates by only training on actual errors.
- Reduce the number of iterations in the iterative SVM

```
GIVEN: DATASET X = (x_1, y_1), ..., (x_n, y_n), C, m, p:

INITIALIZE w := 0, b := 0, SEENDATA := { }
FOR EACH x_i ∈ X DO:
    CLASSIFY x_i USING f(x_i) = sign(< w, x_i > +b)
    IF y_i f(x_i) < m
        FIND w', b' WITH SMO ON SEENDATA,
            USING w, b AS SEED HYPOTHESIS.
        SET (w, b) := (w', b')
    IF SIZE(SEENDATA) > p
        REMOVE OLDEST EXAMPLE FROM SEENDATA
    ADD x_i TO SEENDATA
DONE
```

Figure 4: Pseudo-code for Relaxed Online SVM.

solver by allowing an approximate solution to the optimization problem.

As we describe in the remainder of this subsection, all of these methods trade statistical robustness for reduced computational cost. Experimental results reported in the following section show that they equal or approach the performance of full Online SVMs on content-based spam detection.

3.1 Reducing Problem Size

In the full Online SVMs, we re-optimize over the full set of seen data on every update, which becomes expensive as the number of seen data points grows. We can bound this expense by only considering the p most recent examples for optimization (see Figure 4 for pseudo-code).

Note that this is not equivalent to training a new SVM classifier from scratch on the p most recent examples, because each successive optimization problem is seeded with the previous hypothesis w [8]. This hypothesis may contain values for features that do not occur anywhere in the p most recent examples, and these will not be changed. This allows the hypothesis to remember rare (but informative) features that were learned further than p examples in the past.

Formally, the optimization problem is now defined most clearly in the dual form [23]. In this case, the original soft-margin SVM is computed by maximizing at example n:

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j < \mathbf{x}_i, \mathbf{x}_j >,$$

subject to the previous constraints [23]:

$$\forall i \in \{1, \ldots, n\} : 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

To this, we add the additional lookback buffer constraint

$$\forall j \in \{1, \ldots, (n-p)\} : \alpha_j = c_j$$

where c_j is a constant, fixed as the last value found for α_j while $j > (n-p)$. Thus, the margin found by an optimization is not guaranteed to be one that maximizes the margin for the global data set of examples $\{\mathbf{x}_1, \ldots, \mathbf{x}_n)\}$, but rather one that satisfies a relaxed requirement that the margin be maximized over the examples $\{ \mathbf{x}_{(n-p+1)}, \ldots, \mathbf{x}_n\}$, subject to the fixed constraints on the hyperplane that were found in previous optimizations over examples $\{\mathbf{x}_1, \ldots, \mathbf{x}_{(n-p)}\}$. (For completeness, when $p \geq n$, define $(n-p) = 1$.) This set of constraints reduces the number of free variables in the optimization problem, reducing computational cost.

3.2 Reducing Number of Updates

As noted before, the KKT conditions show that a well classified example will not change the hypothesis; thus it is not necessary to re-train when we encounter such an example. Under the KKT conditions, an example $\mathbf{x}_i$ is considered well-classified when $y_i f(\mathbf{x}_i) > 1$. If we re-train on every example that is not well-classified, our hyperplane will be guaranteed to be optimal at every step.

The number of re-training updates can be reduced by relaxing the definition of *well classified*. An example $\mathbf{x}_i$ is now considered well classified when $y_i f(\mathbf{x}_i) > M$, for some $0 \leq M \leq 1$. Here, each update still produces an optimal hyperplane. The learner may encounter an example that lies within the margins, but farther from the margins than M. Such an example means the hypothesis is no longer globally optimal for the data set, but it is considered good enough for continued use without immediate retraining.

This update procedure is similar to that used by variants of the Perceptron algorithm [18]. In the extreme case, we can set $M = 0$, which creates a mistake driven Online SVM. In the experimental section, we show that this version of Online SVMs, which updates only on actual errors, does not significantly degrade performance on content-based spam detection, but does significantly reduce cost.

3.3 Reducing Iterations

As an iterative solver, SMO makes repeated passes over the data set to optimize the objective function. SMO has one main loop, which can alternate between passing over the entire data set, or the smaller *active set* of current support vectors [22]. Successive iterations of this loop bring the hyperplane closer to an optimal value. However, it is possible that these iterations provide less benefit than their expense justifies. That is, a close first approximation may be good enough. We introduce a parameter T to control the maximum number of iterations we allow. As we will see in the experimental section, this parameter can be set as low as 1 with little impact on the quality of results, providing computational savings.

4. EXPERIMENTS

In Section 2, we argued that the strong performance on content-based spam detection with SVMs with a high value of C show that the maximum margin criteria is overkill, incurring unnecessary computational cost. In Section 3, we proposed ROSVM to address this issue, as both of these methods trade away guarantees on the maximum margin hyperplane in return for reduced computational cost. In this section, we test these methods on the same benchmark data sets to see if state of the art performance may be achieved by these less costly methods. We find that ROSVM is capable of achieving these high levels of performance with greatly reduced cost. Our main tests on content-based spam detection are performed on large benchmark sets of email data. We then apply these methods on the smaller data sets of blog comment spam and blogs, with similar performance.

4.1 ROSVM Tests

In Section 3, we proposed three approaches for reducing the computational cost of Online SMO: reducing the prob-

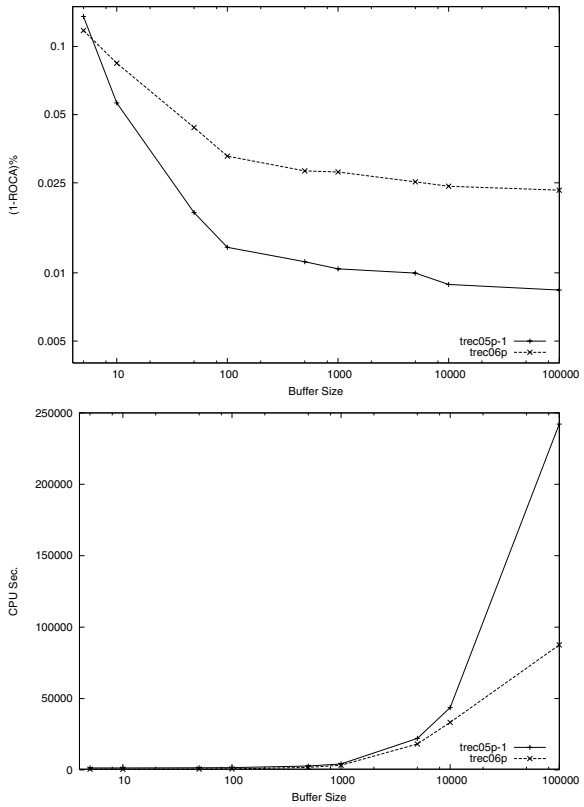

Figure 5: Reduced Size Tests.

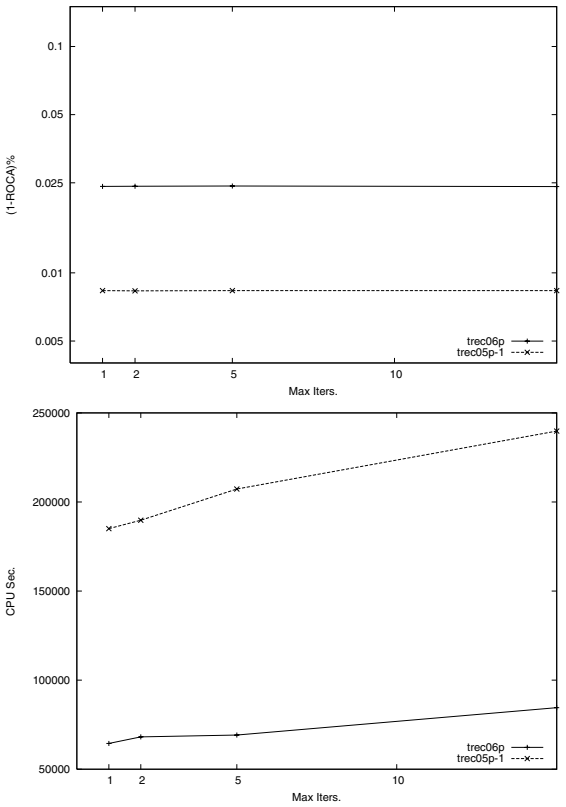

Figure 6: Reduced Iterations Tests.

lem size, reducing the number of optimization iterations, and reducing the number of training updates. Each of these approaches relax the maximum margin criteria on the global set of previously seen data. Here we test the effect that each of these methods has on both effectiveness and efficiency. In each of these tests, we use the large benchmark email data sets, `trec05p-1` and `trec06p`.

4.1.1 Testing Reduced Size

For our first ROSVM test, we experiment on the effect of reducing the size of the optimization problem by only considering the p most recent examples, as described in the previous section. For this test, we use the same 4-gram mappings as for the reference experiments in Section 2, with the same value $C = 100$. We test a range of values p in a coarse grid search. Figure 5 reports the effect of the buffer size p in relationship to the (1-ROCA)% performance measure (top), and the number of CPU seconds required (bottom).

The results show that values of $p < 100$ do result in degraded performance, although they evaluate very quickly. However, p values from 500 to 10,000 perform almost as well as the original Online SMO (represented here as $p = 100,000$), at dramatically reduced computational cost.

These results are important for making state of the art performance on large-scale content-based spam detection practical with online SVMs. Ordinarily, the training time would grow quadratically with the number of seen examples. However, fixing a value of p ensures that the training time is *independent* of the size of the data set. Furthermore, a lookback buffer allows the filter to adjust to *concept drift*.

4.1.2 Testing Reduced Iterations

In the second ROSVM test, we experiment with reducing the number of iterations. Our initial tests showed that the maximum number of iterations used by Online SMO was rarely much larger than 10 on content-based spam detection; thus we tested values of $T = \{1, 2, 5, \infty\}$. Other parameters were identical to the original Online SVM tests.

The results on this test were surprisingly stable (see Figure 6). Reducing the maximum number of SMO iterations per update had essentially no impact on classification performance, but did result in a moderate increase in speed. This suggests that any additional iterations are spent attempting to find improvements to a hyperplane that is already very close to optimal. These results show that for content-based spam detection, we can reduce computational cost by allowing only a single SMO iteration (that is, $T = 1$) with effectively equivalent performance.

4.1.3 Testing Reduced Updates

For our third ROSVM experiment, we evaluate the impact of adjusting the parameter M to reduce the total number of updates. As noted before, when $M = 1$, the hyperplane is globally optimal at every step. Reducing M allows a slightly inconsistent hyperplane to persist until it encounters an example for which it is *too* inconsistent. We tested values of M from 0 to 1, at increments of 0.1. (Note that we used $p = 10000$ to decrease the cost of evaluating these tests.)

The results for these tests are appear in Figure 7, and show that there is a slight degradation in performance with reduced values of M, and that this degradation in performance is accompanied by an increase in efficiency. Values of

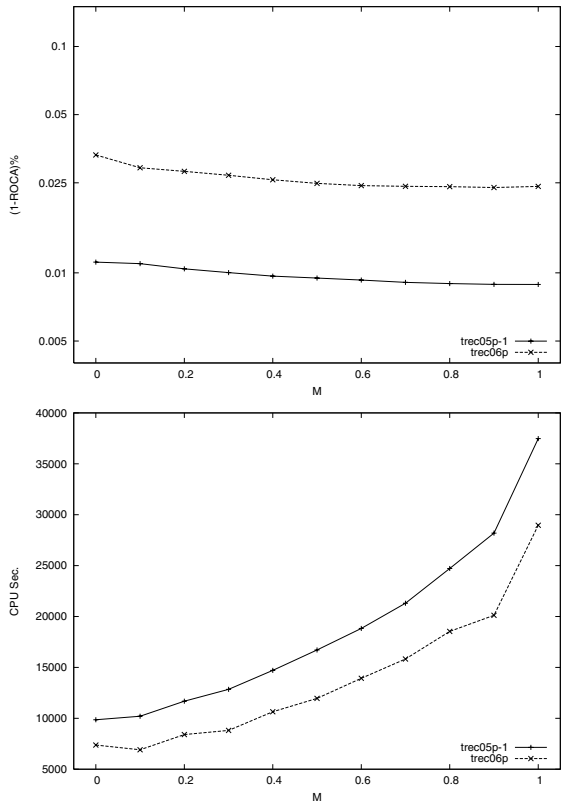

Figure 7: Reduced Updates Tests.

$M > 0.7$ give effectively equivalent performance as $M = 1$, and still reduce cost.

4.2 Online SVMs and ROSVM

We now compare ROSVM against Online SVMs on the email spam, blog comment spam, and splog detection tasks. These experiments show comparable performance on these tasks, at radically different costs. In the previous section, the effect of the different relaxation methods was tested separately. Here, we tested these methods together to create a full implementation of ROSVM. We chose the values $p = 10000$, $T = 1$, $M = 0.8$ for the email spam detection tasks. Note that these parameter values were selected as those allowing ROSVM to achieve comparable performance results with Online SVMs, in order to test total difference in computational cost. The splog and blog data sets were much smaller, so we set $p = 100$ for these tasks to allow meaningful comparisons between the reduced size and full size optimization problems. Because these values were not hand-tuned, both generalization performance and runtime results are meaningful in these experiments.

4.2.1 Experimental Setup

We compared Online SVMs and ROSVM on email spam, blog comment spam, and splog detection. For the email spam, we used the two large benchmark corpora, `trec05p-1` and `trec06p`, in the standard online ordering. We randomly ordered both the blog comment spam corpus and the splog corpus to create online learning tasks. Note that this is a different setting than the leave-one-out cross validation task presented on these corpora in Section 2 – the results are not directly comparable. However, this experimental design does allow meaningful comparison between our two online methods on these content-based spam detection tasks.

We ran each method on each task, and report the results in Tables 5, 6, and 7. Note that the CPU time reported for each method was generated on the same computing system. This time reflects only the time needed to complete online learning on tokenized data. We do not report the time taken to tokenize the data into binary 4-grams, as this is the same additive constant for all methods on each task. In all cases, ROSVM was significantly less expensive computationally.

Table 5: Email Spam Benchmark Data. These results compare Online SVM and ROSVM on email spam detection, using binary 4-gram feature space. Score reported is (1-ROCA)%, where 0 is optimal.

	TREC05P-1 (1-ROC)%	TREC05P-1 CPUs	TREC06P (1-ROC)%	TREC06P CPUs
ONSVM	0.0084	242,160	0.0232	87,519
ROSVM	0.0090	24,720	0.0240	18,541

Table 6: Blog Comment Spam. These results comparing Online SVM and ROSVM on blog comment spam detection using binary 4-gram feature space.

	ACC.	PREC.	RECALL	F1	CPUS
ONSVM	0.926	0.930	0.962	0.946	139
ROSVM	0.923	0.925	0.965	0.945	11

4.3 Discussion

The comparison results shown in Tables 5, 6, and 7 are striking in two ways. First, they show that the performance of Online SVMs can be matched and even exceeded by relaxed margin methods. Second, they show a dramatic disparity in computational cost. ROSVM is an order of magnitude more efficient than the normal Online SVM, and gives comparable results. Furthermore, the fixed lookback buffer ensures that the cost of each update does not depend on the size of the data set already seen, unlike Online SVMs. Note the blog and splog data sets are relatively small, and results on these data sets must be considered preliminary. Overall, these results show that there is no need to pay the high cost of SVMs to achieve this level of performance on content-based detection of spam. ROSVMs offer a far cheaper alternative with little or no performance loss.

5. CONCLUSIONS

In the past, academic researchers and industrial practitioners have disagreed on the best method for online content-based detection of spam on the web. We have presented one resolution to this debate. Online SVMs do, indeed, pro-

Table 7: Splog Data Set. These results compare Online SVM and ROSVM on splog detection using binary 4-gram feature space.

	ACC.	PREC.	RECALL	F1	CPUS
ONSVM	0.880	0.910	0.842	0.874	29353
ROSVM	0.878	0.902	0.849	0.875	1251

duce state-of-the-art performance on this task with proper adjustment of the tradeoff parameter C, but with cost that grows quadratically with the size of the data set. The high values of C required for best performance with SVMs show that the margin maximization of Online SVMs is overkill for this task. Thus, we have proposed a less expensive alternative, ROSVM, that relaxes this maximum margin requirement, and produces nearly equivalent results. These methods are efficient enough for large-scale filtering of content-based spam in its many forms.

It is natural to ask *why* the task of content-based spam detection gets strong performance from ROSVM. After all, not all data allows the relaxation of SVM requirements. We conjecture that email spam, blog comment spam, and splogs all share the characteristic that a subset of features are particularly indicative of content being either spam or not spam. These indicative features may be sparsely represented in the data set, because of spam methods such as word obfuscation, in which common spam words are intentionally misspelled in an attempt to reduce the effectiveness of word-based spam detection. Maximizing the margin may cause these sparsely represented features to be ignored, creating an overall reduction in performance. It appears that spam data is highly separable, allowing ROSVM to be successful with high values of C and little effort given to maximizing the margin. Future work will determine how applicable relaxed SVMs are to the general problem of text classification.

Finally, we note that the success of relaxed SVM methods for content-based spam detection is a result that depends on the nature of spam data, which is potentially subject to change. Although it is currently true that ham and spam are linearly separable given an appropriate feature space, this assumption may be subject to attack. While our current methods appear robust against primitive attacks along these lines, such as the *good word attack* [24], we must explore the feasibility of more sophisticated attacks.

6. REFERENCES

[1] A. Bratko and B. Filipic. Spam filtering using compression models. Technical Report IJS-DP-9227, Department of Intelligent Systems, Jozef Stefan Institute, L jubljana, Slovenia, 2005.

[2] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *NIPS*, pages 409–415, 2000.

[3] G. V. Cormack. TREC 2006 spam track overview. In *To appear in: The Fifteenth Text REtrieval Conference (TREC 2006) Proceedings*, 2006.

[4] G. V. Cormack and A. Bratko. Batch and on-line spam filter comparison. In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS)*, 2006.

[5] G. V. Cormack and T. R. Lynam. TREC 2005 spam track overview. In *The Fourteenth Text REtrieval Conference (TREC 2005) Proceedings*, 2005.

[6] G. V. Cormack and T. R. Lynam. On-line supervised spam filter evaluation. Technical report, David R. Cheriton School of Computer Science, University of Waterloo, Canada, February 2006.

[7] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines*. Cambridge University Press, 2000.

[8] D. DeCoste and K. Wagstaff. Alpha seeding for support vector machines. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 345–349, 2000.

[9] H. Drucker, V. Vapnik, and D. Wu. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

[10] J. Goodman and W. Yin. Online discriminative spam filter training. In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS)*, 2006.

[11] P. Graham. A plan for spam. 2002.

[12] P. Graham. Better bayesian filtering. 2003.

[13] Z. Gyongi and H. Garcia-Molina. Spam: It's not just for inboxes anymore. *Computer*, 38(10):28–34, 2005.

[14] T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, 1998.

[15] T. Joachims. Training linear svms in linear time. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, 2006.

[16] J. Kivinen, A. Smola, and R. Williamson. Online learning with kernels. In *Advances in Neural Information Processing Systems 14*, pages 785–793. MIT Press, 2002.

[17] P. Kolari, T. Finin, and A. Joshi. SVMs for the blogosphere: Blog identification and splog detection. *AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs*, 2006.

[18] W. Krauth and M. Mézard. Learning algorithms with optimal stability in neural networks. *Journal of Physics A*, 20(11):745–752, 1987.

[19] T. Lynam, G. Cormack, and D. Cheriton. On-line spam filter fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 123–130, 2006.

[20] V. Metsis, I. Androutsopoulos, and G. Paliouras. Spam filtering with naive bayes – which naive bayes? *Third Conference on Email and Anti-Spam (CEAS)*, 2006.

[21] G. Mishne, D. Carmel, and R. Lempel. Blocking blog spam with language model disagreement. *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, May 2005.

[22] J. Platt. Sequenital minimal optimization: A fast algorithm for training support vector machines. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[23] B. Scholkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[24] G. L. Wittel and S. F. Wu. On attacking statistical spam filters. *CEAS: First Conference on Email and Anti-Spam*, 2004.