A Sequential Dual Method for Large Scale Multi-Class Linear SVMs

S. Sathiya Keerthi
Yahoo! Research
Santa Clara, CA
selvarak@yahoo-inc.com

S. Sundararajan
Yahoo! Labs
Bangalore, India
ssrajan@yahoo-inc.com

Kai-Wei Chang
Dept. of Computer Science
National Taiwan University
Taipei 106, Taiwan
b92084@csie.ntu.edu.tw

Cho-Jui Hsieh
Dept. of Computer Science
National Taiwan University
Taipei 106, Taiwan
b92085@csie.ntu.edu.tw

Chih-Jen Lin
Dept. of Computer Science
National Taiwan University
Taipei 106, Taiwan
cjlin@csie.ntu.edu.tw

ABSTRACT

Efficient training of direct multi-class formulations of linear Support Vector Machines is very useful in applications such as text classification with a huge number examples as well as features. This paper presents a fast dual method for this training. The main idea is to sequentially traverse through the training set and optimize the dual variables associated with one example at a time. The speed of training is enhanced further by shrinking and cooling heuristics. Experiments indicate that our method is much faster than state of the art solvers such as bundle, cutting plane and exponentiated gradient methods.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*

General Terms

Algorithms, Performance, Experimentation

1. INTRODUCTION

Support vector machines (SVM) [2] are popular methods for solving classification problems. An SVM employing a nonlinear kernel maps the input $\boldsymbol{x}$ to a high dimensional feature space via a nonlinear function $\phi(\boldsymbol{x})$ and forms a linear classifier in that space to solve the problem. Inference as well as design are carried out using the kernel function, $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$. In several data mining applications the input vector $\boldsymbol{x}$ itself has rich features and so it is sufficient to take ϕ as the identity map, i.e. $\phi(\boldsymbol{x}) = \boldsymbol{x}$. SVMs developed in this setting are referred to as Linear SVMs.

KDD'08, August 24–27, 2008, Las Vegas, Nevada, USA.

In applications such as text classification the input vector $\boldsymbol{x}$ resides in a high dimensional space (e.g. a bag-of-words representation), the training set has a large number of labeled examples, and the training data matrix is sparse, i.e. the average number of non-zero elements in one $\boldsymbol{x}$ is small. Most classification problems arising in such domains involve many classes. Efficient learning of the parameters of *linear* SVMs for such problems is important. The One-Versus-Rest (OVR) method which consists of developing, for each class, a binary classifier that separates that class from the rest of the classes, and then combining the classifiers for multi-class inference, is one of the popular schemes to solve a multi-class problem. OVR is easy and efficient to design and also gives good performance [21].

Crammer and Singer [5, 6] and Weston and Watkins [25] gave direct multi-class SVM formulations. With nonlinear kernels in mind, Crammer and Singer [5, 6] and Hsu and Lin [10] gave efficient decomposition methods for these formulations. Recently, some good methods have been proposed for SVMs with structured outputs, that also specialize nicely for Crammer-Singer multi-class linear SVMs. Teo et al. [23] suggested a bundle method and Joachims et al. [13] gave a cutting plane method that is very close to it; these methods can be viewed as extensions of the method given by Joachims [12] for binary linear SVMs. Collins et al. [4] gave the exponentiated gradient method.

In this paper we present fast dual methods for Crammer-Singer and Weston-Watkins formulations. Like the methods of Crammer and Singer [5, 6] and Hsu and Lin [10] our methods optimize the dual variables associated with one example at a time. However, while those previous methods always choose the example that violates optimality the most for updating, our methods sequentially traverse through the examples. For the linear SVM case this leads to significant differences. In particular, for this case our methods are extremely efficient while those methods are not so efficient. We borrow and modify the shrinking and cooling heuristics of Crammer and Singer [6] to make our methods even faster. Experiments on several datasets indicate that our method is also much faster than cutting plane, bundle and exponentiated gradient methods. For the special case of binary classification our methods turn out to be equivalent to doing coordinate descent. We covered this method in detail in [9].

Table 1.1: Properties of datasets

Dataset	l	l_{test}	n	k	# nonzeros in whole data
NEWS20	15,935	3,993	62,061	20	1,593,692
SECTOR	6,412	3,207	55,197	105	1,569,904
MNIST	60,000	10,000	779	10	10,505,375
RCV1	531,742	15,913	47,236	103	36,734,621
COVER	522,911	58,101	54	7	6,972,144

Notations. The following notations are used in the paper. We use l to denote the number of training examples, l_{test} to denote the number of test examples and k to denote the number of classes. Throughout the paper, the index i will denote a training example and the index m will denote a class. Unless otherwise mentioned, i will run from 1 to l and m will run from 1 to k. $\boldsymbol{x}_i \in R^n$ is the input vector (n is the number of input features) associated with the i-th example and $y_i \in \{1, \ldots, k\}$ is the corresponding target class. We will use $\mathbf{1}$ to denote a vector of all 1's whose dimension will be clear from the context. Superscript T will denote transpose for vectors.

The following multi-class datasets are used in the paper for doing various experiments: NEWS20 [14], SECTOR [20, 19], MNIST [15], RCV1 [16] and COVER [13]. RCV1 is a taxonomy based multi-label dataset and so we modified it to a multi-class dataset by removing all examples which had mismatched labels in the taxonomy tree; in the process of doing this, two classes do not have any training examples. For NEWS20, MNIST and RCV1 we used the standard train-test split; for SECTOR we used the second train/test split used in [20]; and, for COVER we used the train/test split used in [13]. Table 1.1 gives key properties of these datasets.

The paper is organized as follows. In section 2 we give the details of our method (we call it as the *Sequential Dual Method* (SDM)) for the Crammer-Singer formulation and evaluate it against other methods. In section 3 we describe SDM for the Weston-Watkins formulation. In section 4 we compare these methods with the One-Versus-Rest method of solving multi-class problems. We conclude the paper in section 5. An implementation of a variant of the SDM algorithm for Crammer-Singer formulation is available in the LIBLINEAR library (version 1.3 or higher) with the option `-s 4` (`http://www.csie.ntu.edu.tw/~cjlin/liblinear`).

2. SDM FOR CRAMMER-SINGER FORMULATION

In [5, 6], Crammer and Singer proposed an approach for the multi-class problem by formulating the following primal problem:

$$\begin{aligned} \min_{\{\boldsymbol{w}_m\},\{\xi_i\}} \quad & \tfrac{1}{2}\textstyle\sum_m \|\boldsymbol{w}_m\|^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & \boldsymbol{w}_{y_i}^T\boldsymbol{x}_i - \boldsymbol{w}_m^T\boldsymbol{x}_i \geq e_i^m - \xi_i \ \ \forall m, i, \end{aligned} \tag{1}$$

where $C > 0$ is the regularization parameter, $\boldsymbol{w}_m$ is the weight vector associated with class m, $e_i^m = 1 - \delta_{y_i,m}$ and $\delta_{y_i,m} = 1$ if $y_i = m$, $\delta_{y_i,m} = 0$ if $y_i \neq m$. Note that, in (1) the constraint for $m = y_i$ corresponds to the non-negativity constraint, $\xi_i \geq 0$. The decision function is

$$\arg\max_m \ \boldsymbol{w}_m^T\boldsymbol{x}. \tag{2}$$

The dual problem of (1), developed along the lines of [5, 6], involves a vector $\boldsymbol{\alpha}$ having dual variables $\alpha_i^m \ \forall m, i$. The $\boldsymbol{w}_m$ get defined via $\boldsymbol{\alpha}$ as

$$\boldsymbol{w}_m(\boldsymbol{\alpha}) = \sum_i \alpha_i^m \boldsymbol{x}_i \ \ \forall\, m \tag{3}$$

At most places below, we simply write $\boldsymbol{w}_m(\boldsymbol{\alpha})$ as $\boldsymbol{w}_m$. Let $C_i^m = 0$ if $y_i \neq m$, $C_i^m = C$ if $y_i = m$. The dual problem is

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) = \tfrac{1}{2}\textstyle\sum_m \|\boldsymbol{w}_m(\boldsymbol{\alpha})\|^2 + \sum_i \sum_m e_i^m \alpha_i^m \\ \text{s.t.} \quad & (\alpha_i^m \leq C_i^m \ \forall m, \ \ \textstyle\sum_m \alpha_i^m = 0) \ \ \forall i, \end{aligned} \tag{4}$$

The gradient of f plays an important role and is given by

$$g_i^m = \frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_i^m} = \boldsymbol{w}_m(\boldsymbol{\alpha})^T\boldsymbol{x}_i + e_i^m \ \ \forall\, i, m. \tag{5}$$

If $\bar{n}$ is the average number of nonzero elements per training example, then on average the evaluation of each g_i^m takes $O(\bar{n})$ effort. Optimality of $\boldsymbol{\alpha}$ for (4) can be checked using the quantity,

$$v_i = \max_m g_i^m - \min_{m:\alpha_i^m < C_i^m} g_i^m \ \ \forall i. \tag{6}$$

For a given i, the values of m that attain the max and min in (6) play a useful role and are denoted as follows:

$$M_i = \arg\max_m \ g_i^m \ \text{ and } \ m_i = \arg\min_{m:\alpha_i^m < C_i^m} g_i^m. \tag{7}$$

From (6) it is clear that v_i is non-negative. Dual optimality holds when:

$$v_i = 0 \ \ \forall i. \tag{8}$$

For practical termination we can approximately check optimality using a tolerance parameter, $\epsilon > 0$:

$$v_i < \epsilon \ \ \forall i. \tag{9}$$

We will also refer to this as ϵ-optimality. The value $\epsilon = 0.1$ is a good choice for a practical implementation of SDM.

SDM consists of sequentially picking one i at a time and solving the restricted problem of optimizing only $\alpha_i^m \ \forall m$, keeping all other variables fixed.* To do this, we let $\delta\alpha_i^m$ denote the additive change to be applied to the current α_i^m, and optimize $\delta\alpha_i^m \ \forall m$. Let $\boldsymbol{\alpha}_i$, $\boldsymbol{\delta\alpha}_i$, $\boldsymbol{g}_i$ and $\boldsymbol{C}_i$ be vectors that respectively gather the elements α_i^m, $\delta\alpha_i^m$, g_i^m and C_i^m over all m. With $A_i = \|\boldsymbol{x}_i\|^2$ the sub-problem of optimizing $\boldsymbol{\delta\alpha}_i$ is given by

$$\begin{aligned} \min_{\boldsymbol{\delta\alpha}_i} \quad & \tfrac{1}{2}A_i\|\boldsymbol{\delta\alpha}_i\|^2 + \boldsymbol{g}_i^T\boldsymbol{\delta\alpha}_i \\ \text{s.t.} \quad & \boldsymbol{\delta\alpha}_i \leq \boldsymbol{C}_i - \boldsymbol{\alpha}_i, \ \ \mathbf{1}^T\boldsymbol{\delta\alpha}_i = 0 \end{aligned} \tag{10}$$

This can be derived by noting the following: (i) changing α_i^m to $\alpha_i^m + \delta\alpha_i^m$ causes $\boldsymbol{w}_m$ to change to $\boldsymbol{w}_m + \delta\alpha_i^m\boldsymbol{x}_i$; (ii) $\|\boldsymbol{w}_m + \delta\alpha_i^m\boldsymbol{x}_i\|^2 = \|\boldsymbol{w}_m\|^2 + A_i(\delta\alpha_i^m)^2 + 2(\boldsymbol{w}_m^T\boldsymbol{x}_i)\delta\alpha_i^m$; (iii) $e_i^m\alpha_i^m$ changes to $e_i^m\alpha_i^m + e_i^m\delta\alpha_i^m$; (iv) using the definition

*When specialized to binary classification ($k = 2$), for each i the equality constraint in (4) can be used to eliminate the variable α_i^m, $m \neq y^i$. With this done and the dual solution viewed as optimization for the variables $\{\alpha_i^{y_i}\}_i$, SDM is equivalent to doing coordinate descent. This is what we did in [9].

Algorithm 2.1 SDM for Crammer-Singer Formulation

- Initialize $\boldsymbol{\alpha}$ and the corresponding $\boldsymbol{w}_m \ \forall m$.
- Until (8) holds in an entire loop over examples, do:

  For $i = 1, \ldots, l$

  (a) Compute $g_i^m \ \forall m$ and obtain v_i

  (b) If $v_i \neq 0$, solve (10) and set:

  $$\boldsymbol{\alpha}_i \leftarrow \boldsymbol{\alpha}_i + \boldsymbol{\delta\alpha}_i$$
  $$\boldsymbol{w}_m \leftarrow \boldsymbol{w}_m + \delta\alpha_i^m \boldsymbol{x}_i \ \forall m$$

of g_i^m in (5); and (v) using the above in (4) and leaving out all constants that do not depend on $\boldsymbol{\delta\alpha}_i$. The sub-problem in (10) has a nice simple form. Let us discuss methods for solving it. Suppose $A_i = 0$ for some i, which can happen only when $\boldsymbol{x}_i = \mathbf{0}$. In the primal problem (1) such examples contribute a fixed cost of C to the objective function, they have no effect on the $\boldsymbol{w}_m$, and so they can be effectively ignored. *So, hereafter we will assume $A_i > 0 \ \forall i$.* Crammer and Singer [5, 6] suggest two methods for solving (10): (i) an exact $O(k \log k)$ algorithm (see section 6.2 of [5]) and (ii) an approximate iterative fixed-point algorithm (see section 6 of [6]). Alternatively, one could also employ an active set method meant for convex quadratic programming problems [8], starting from $\boldsymbol{\delta\alpha}_i = \mathbf{0}$ as the initial point. Since the Hessian of the objective function in (10) is A_i times the identity matrix and the constraints of (10) are in a simple form, various linear equation solving steps of the active set method can be done analytically and cheaply. Our implementation uses this method.

A description of SDM is given in Algorithm 2.1. If good seed values are unavailable, a simple way to initialize is to set $\boldsymbol{\alpha} = \mathbf{0}$; this corresponds to $\boldsymbol{w}_m = \mathbf{0} \ \forall m$. Let us now discuss the complexity of the algorithm. Each execution of step (a) requires $O(k\bar{n})$ effort; recall that $\bar{n}$ is the average number of nonzero elements per training example. The cost of step (b) is at most $O(k\bar{n})$, and it could be much less depending on the number of α_i^m that are actually changed during the solution of (10). Since the solution of (10) itself is usually not as high as these costs if k is not very large, it is reasonable to take the cost of an update for one i as $O(k\bar{n})$. Thus the overall cost of a loop, i.e. a full sequence of updates over all examples is $O(lk\bar{n})$. The main memory requirement of the algorithm consists of storing the data, $\boldsymbol{x}_i \ \forall i$ ($O(l\bar{n})$), the weights, $\boldsymbol{w}_m \ \forall m$ ($O(kn)$) and the α_i^m $\forall i, m$ (at most $O(kl)$). For problems with a large number of classes but having only a small number of non-zero α_i^m, it is prudent to use a linked list data structure and store only those α_i^m which are non-zero.

The following convergence theorem can be proved for SDM, employing ideas similar to those in [17].

Theorem 2.1 *Let $\boldsymbol{\alpha}^t$ denote the $\boldsymbol{\alpha}$ at the end of the t-th loop of Algorithm 2.1. Any limit point of a convergent subsequence of $\{\boldsymbol{\alpha}^t\}$ is a global minimum of (4).*

Because of space limitation we omit the proof. In [9] we showed that SDM for binary classification has a linear rate of convergence, i.e., there exists $0 \leq \mu < 1$ and a t_0 such that

$$f(\boldsymbol{\alpha}^t) - f(\boldsymbol{\alpha}^*) \leq \mu^{t-t_0}(f(\boldsymbol{\alpha}^{t_0}) - f(\boldsymbol{\alpha}^*)), \forall t \geq t_0, \quad (11)$$

where f is the dual objective function, t is the loop count in the algorithm, $\boldsymbol{\alpha}^t$ is the $\boldsymbol{\alpha}$ at the end of the t-th loop, and $\boldsymbol{\alpha}^*$ is the dual optimal solution. On the other hand Theorem 2.1 only implies a weaker form of convergence for Algorithm 2.1. As we remarked earlier, for binary classification SDM becomes coordinate descent, a standard method for which good convergence results exist in the optimization literature [18]. The presence of the equality constraint in (4) makes the multi-class version somewhat non-standard. It is an interesting open problem to establish linear convergence for Algorithm 2.1.

Let us now discuss various ways of making Algorithm 2.1 more efficient. In the algorithm the examples are considered in the given order $i = 1, 2, \ldots, l$ for updating. Any systematic regularities in the given order (for instance, all examples of one class coming together) may lead to slow convergence. So it is a good idea to randomly permute the examples. In fact it is useful to do this random permutation before each loop through the examples. We do this in our implementation of SDM. In [9] we found the same idea to be useful for binary classification. Theorem 2.1 is valid even in the presence of such random permutations.

In the solution of the sub-problem (10), the variables, α_i^m corresponding to the 'most violating' indices $m \in \{M_i, m_i\}$ (see (7)) are particularly important. When the number of classes is large, for efficiency one could consider solving the sub-problem only for these two variables. An extended version of this idea is the following. Since the inactive variables, i.e., $m : \alpha_i^m < C_i^m$ are important, we can optimize on the variables α_i^m for $m = M_i$ and $m : \alpha_i^m < C_i^m$. In most problems the number of inactive variables per example is quite small[†] (even when the number of classes is large) and so we use this method in our implementation.

2.1 Heuristics for improving efficiency

Crammer and Singer [6] employed two heuristics to speed up their algorithm. We employ these heuristics for SDM too. We now discuss these heuristics.

Shrinking. Shrinking [11], a technique for removing variables that are not expected to change, is known to be a very good way to speed up decomposition methods for binary SVMs. In the multi-class case, it may turn out that for many examples $\alpha_i^m = 0 \ \forall m$. So, for efficiency Crammer and Singer [6] suggested the idea of concentrating the algorithm's effort on examples i for which there is at least one non-zero α_i^m. (If, for a given i there is at least one non-zero α_i^m, then it is useful to observe the following using the special nature of the constraints in the dual problem: (i) there are at least two non-zero α_i^m; and (ii) $\alpha_i^{y_i} > 0$.) We can get further improved efficiency by doing two modifications. The first modification is to also remove those i for which there is exactly one $m \neq y_i$ with $\alpha_i^m = -\alpha_i^{y_i} = -C$. (In other words, apart from removing those i with $\alpha_i^m = 0$ $\forall m$, we are also removing those i with exactly two non-zero α_i^m each having magnitude C.) At optimality this *generically* corresponds to $\arg\max_{m \neq y_i} \boldsymbol{w}_m^T \boldsymbol{x}_i$ being a singleton and $\xi_i = \boldsymbol{w}_m^T \boldsymbol{x}_i - \boldsymbol{w}_{y_i}^T \boldsymbol{x}_i + 1 > 0$. As we approach optimality, for the m that attains the arg max above we expect α_i^m to stay at $-C$, $\alpha_i^{y_i}$ to stay at C and $\alpha_i^m = 0$ for all other m. If there are many such examples, then this modification will give good benefits. The second modification is the follow-

[†]This corresponds to the observation that *for any given example* the confusion in classification is only amongst a few classes.

ing: even within an example i that does not get removed, consider for updating only those α_i^m which are non-zero. Efficiency improvement due to this modification can be substantial when the average number of non-zero α_i^m for each example is much smaller than the number of classes. With these modifications included in shrinking SDM is changed as follows. We alternate between the following two steps.

1. First do a full sequential loop over all examples.
2. Do several loops over only those examples i that do not satisfy the condition for removal discussed above. When doing this, for each i that is included, evaluate g_i^m and optimize α_i^m only for those m that have non-zero α_i^m. Repeat these 'shrunk' loops until either ϵ-optimality is satisfied on all the positive α_i^m or the effort associated with these loops exceeds a specified amount.[‡]

Cooling of accuracy parameter. Suppose we desire to solve the dual problem quite accurately using a small value of ϵ. When shrinking is employed there is a tendency for the algorithm to spend a lot of effort in the 'shrunk' loops. The algorithm has to wait for the next full loop (where other optimality-violating α_i^m get included) to make a decent improvement in the dual objective function. To avoid a staircase behavior of the objective function with computing time, it is a good idea to start with a large value for ϵ and decrease it in several graded steps until the final desired ϵ is reached. In our implementation we start from $\epsilon = 1$ and decrease it by a factor of 10 in each graded step.

Figure 1 demonstrates the effectiveness of including the heuristics on two datasets. Clearly, the heuristics are valuable when (4) needs to be solved quite accurately.

2.2 Comparison with Crammer and Singer's method

In [5, 6], Crammer and Singer gave a method for solving (4) with nonlinear kernels in mind. Their method also uses the basic idea of choosing an i and solving (10) to update $\boldsymbol{\alpha}_i$. Unlike our method which does a sequential traversal of the i, Crammer and Singer's method always chooses the i for which the optimality violation v_i in (6) is largest at the current $\boldsymbol{\alpha}$. To appreciate Crammer and Singer's motivation for doing this, first note that the g_i^m are key quantities for calculating the v_i as well as for setting up (10). So let us look at the computational complexity associated with g_i^m in some detail.

For problems with a nonlinear kernel $K(\boldsymbol{x}_j, \boldsymbol{x}_i)$, the g_i^m need to be computed using

$$g_i^m = \sum_j \alpha_j^m K(\boldsymbol{x}_j, \boldsymbol{x}_i) + e_i^m \quad \forall\, i, m. \tag{12}$$

Let us recall the notations $\boldsymbol{g}_i$, $\boldsymbol{\alpha}_i$ and $\boldsymbol{\delta\alpha}_i$ given above (10). Direct computation of $\boldsymbol{g}_i$ for one given i via (12) takes $O(lk\bar{n})$ effort. On the other hand, suppose we maintain $\boldsymbol{g}_j$ $\forall j$. In a typical step of Crammer and Singer's method, an i is chosen and $\boldsymbol{\alpha}_i$ is changed as $\boldsymbol{\alpha}_i \leftarrow \boldsymbol{\alpha}_i + \boldsymbol{\delta\alpha}_i$; in such a case the $\boldsymbol{g}_j$ can be updated using

$$\boldsymbol{g}_j \leftarrow \boldsymbol{g}_j + K(\boldsymbol{x}_j, \boldsymbol{x}_i)\boldsymbol{\delta\alpha}_i \quad \forall\, j. \tag{13}$$

[‡]In our implementation we count the total number of g_i^m evaluations, divide it by kl to get the number of effective loops and terminate the 'shrunk' loops when the number of effective loops exceeds 5.

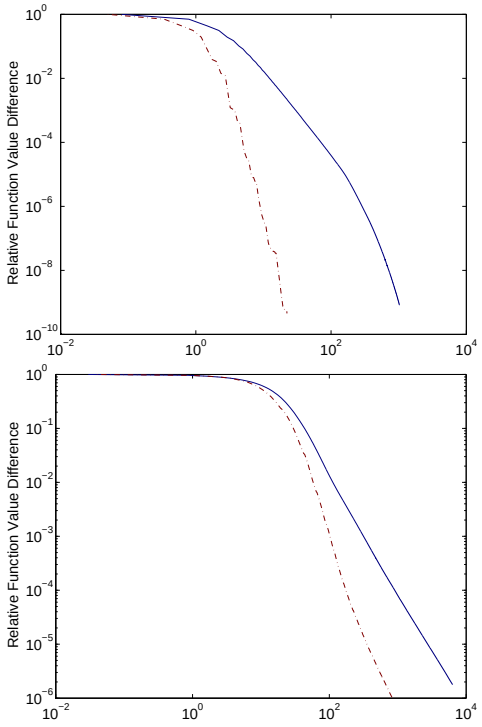

Figure 1: Comparison of *SDM* with (Red, dashdot) and without (Blue, solid) the shrinking and cooling heuristics on MNIST (top) and COVER (bottom) datasets. The horizontal axis is Training time in seconds.

which also requires (total, for all j) only $O(lk\bar{n})$ effort. Therefore maintaining $\boldsymbol{g}_j$ $\forall j$ does not cost more than the direct computation of a single $\boldsymbol{g}_i$ using (12). As using $\boldsymbol{g}_j$ $\forall j$ implies fewer iterations (i.e., faster convergence due to the ability to choose for updating the i that violates optimality most), of course one should take this advantage. This clearly motivates the scheme for choosing i in Crammer and Singer's method.

However, the situation for linear SVM is very different. With the cheap $O(k\bar{n})$ way ($g_i^m = \boldsymbol{w}_m^T \boldsymbol{x}_i + e_i^m$ $\forall m$) of computing $\boldsymbol{g}_i$ and the cheap $O(k\bar{n})$ way ($\boldsymbol{w}_m \leftarrow \boldsymbol{w}_m + \delta\alpha_i^m \boldsymbol{x}_i$ $\forall m$) of updating the $\boldsymbol{w}_m$, each update of $\boldsymbol{\alpha}_i$ takes only $O(k\bar{n})$ effort, and so, if l is large, SDM can do many more ($O(l)$) updates of the $\boldsymbol{w}_m$ for the same effort as needed by Crammer and Singer's method for one update of the $\boldsymbol{w}_m$. Hence, for linear SVM, always choosing the i that yields the maximum v_i is unwise and sequential traversal of the i is a much better choice.

2.3 Comparison with other methods

We now compare our method with other methods for solving (1). The cutting plane method of Joachims et al. [13] and the bundle method of Teo et al. [23] are given for the more general setting of SVMs with structured outputs, but they also specialize to multi-class linear SVMs using the Crammer-Singer formulation. These methods combine the loss term $C\sum_i \xi_i$ in (1) into a single loss function and build convex, piecewise linear lower bounding approximations for this function by using sub-differentials of it from various points encountered during the solution. A negative aspect

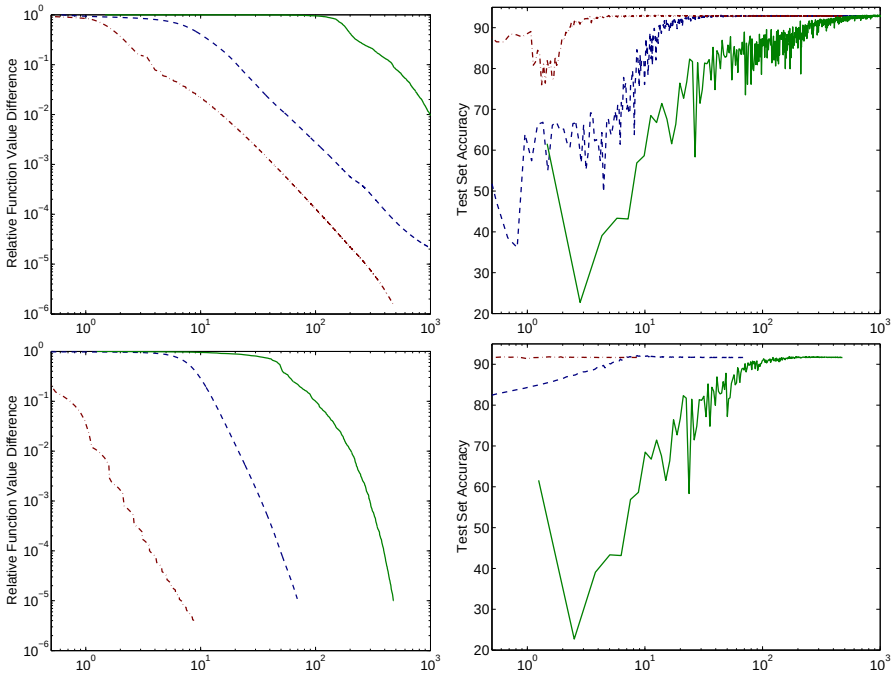

Figure 2: Comparison of *SDM* (Red, dashdot), *EG* (Blue, dashed) and *Bundle* (Green, solid) on MNIST dataset. The horizontal axis is Training time in seconds. Each row corresponds to one C value; Row 1: $C = 10$; Row 2: $C = 0.1$

of these algorithms is their need to scan the entire training set in order to form one sub-differential mentioned above; as we will see below in the experiments, this causes these algorithms to be slow to converge. Barring some differences in implementation the algorithms in [13] and [23] are the same and we will simply refer to them jointly as the Cutting Plane (CP) method. For comparison we use the implementation in `http://www.cs.cornell.edu/People/tj/svm_light/svm_multiclass.html`.

The Exponentiated Gradient (EG) method [4] also applies to structured output problems and specializes to the Crammer-Singer multi-class linear SVM formulation. It uses a dual form that is slightly different from (4), but is equivalent to it by a simple translation and scaling of variables. Like our method the EG method updates the dual variables associated with one example at a time, but the selection of examples is done in an online mode where an i is picked randomly each time. The EG method does an approximate descent via a constraint-obeying exponentiated gradient step. By its very design all dual inequality constraints remain inactive (i.e., $\alpha_i^m < C_i^m \ \forall i, m$) throughout the algorithm, though, as optimality is reached some of them may asymptotically become active. The EG method is very straightforward to implement and we use our own implementation for the experiments given below.

We now compare our method (SDM) against CP and EG on the five datasets used in this paper, both with respect to the ability to converge well to the optimum of (4) as well as the ability to reach steady state test set performance fast. We evaluate the former by the *Relative Function Value Difference* given by $(f - f^\star)/f^\star$ where $f^\star$ is the optimal objective function value of (4), and the latter by *Test Set Accuracy* which is the percentage of test examples that are classified correctly. For simplicity we set $C = 1$ for this experiment. Figure 3 (see the last page of the paper) gives plots of these two quantities as a function of training time for the five datasets. Overall, SDM is much faster than EG, which in turn is much faster than CP. This is true irrespective of whether the aim is to efficiently reach the optimum dual objective very closely or reach the steady state test accuracy fast. On COVER (see the bottom left row of Figure 3) EG has difficulty reaching the optimal objective function value, which is possibly due to numerical issues associated with large exponential operations involved in its updates.

When the regularization parameter C needs to be tuned, say via cross-validation techniques, it is necessary to solve (4) for a range of C values. It is important for an algorithm to be efficient for widely varying C values. For one dataset, MNIST, Figure 2 compares the three methods for large and small C values ($C = 10$ and $C = 0.1$); for the intermediate value of $C = 1$, see also the plots for MNIST in Figure 3. When actually finding solutions for an ordered set of several C values, the optimal $\boldsymbol{\alpha}$ and the $\boldsymbol{w}_m$ obtained from the algorithm for one C value can be used to form initializations of corresponding quantities of the algorithm for the next C value. This is a standard seeding technique that improves efficiency, and is also used in the EG method [4].

Stochastic gradient descent methods [22, 3] can be applied to the online version of (1) and they are known to achieve steady state test performance quite fast. We have not evaluated SDM against these methods. However, our evaluation in [9] shows that, for binary classification SDM is faster than stochastic gradient methods. So we expect the same result to hold for the multi-class case too. The modified MIRA algorithm proposed by Crammer and Singer [7] has some similarities with our sequential dual method, but it is given in an online setting and does not solve the batch problem. Also, ideas such as shrinking do not apply to it. Bordes et al. [1] apply a coordinate descent method for multi-class SVM, but they focus on nonlinear kernels.

3. SDM FOR WESTON-WATKINS FORMULATION

Let e_i^m and δ_i^m be as defined in section 2. In [25], Weston and Watkins proposed an approach for the multi-class problem by formulating the following primal problem:

$$\min_{\{\boldsymbol{w}_m\},\{\xi_i^m\}} \frac{1}{2}\sum_m \|\boldsymbol{w}_m\|^2 + C\sum_i \sum_{m\neq y_i} \xi_i^m$$

$$\text{s.t. } \boldsymbol{w}_{y_i}^T\boldsymbol{x}_i - \boldsymbol{w}_m^T\boldsymbol{x}_i \geq e_i^m - \xi_i^m,\ \ \xi_i^m \geq 0\ \ \forall m \neq y_i,\ \forall i. \quad (14)$$

The decision function remains the same as in (2). The dual problem of (14) involves a vector $\boldsymbol{\alpha}$ having dual variables $\alpha_i^m\ \forall i, m \neq y_i$. For convenience let us set

$$\alpha_i^{y_i} = -\sum_{m\neq y_i} \alpha_i^m,$$

and define

$$\boldsymbol{w}_m(\boldsymbol{\alpha}) = -\sum_i \alpha_i^m \boldsymbol{x}_i. \quad (15)$$

The dual of (14) can now be written as

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) = \tfrac{1}{2}\textstyle\sum_m \|\boldsymbol{w}_m(\boldsymbol{\alpha})\|^2 + \sum_i \sum_{m\neq y_i} \alpha_i^m \\ \text{s.t.} \quad & 0 \leq \alpha_i^m \leq C\ \ \forall m \neq y_i,\ \ \forall i. \end{aligned} \quad (16)$$

The gradient of f is given by

$$g_i^m = \frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_i^m} = \boldsymbol{w}_{y_i}(\boldsymbol{\alpha})^T\boldsymbol{x}_i - \boldsymbol{w}_m(\boldsymbol{\alpha})^T\boldsymbol{x}_i - 1\ \ \forall\ i,\ m \neq y_i. \quad (17)$$

Optimality can be checked using v_i^m, $m \neq y_i$, defined as:

$$v_i^m = \begin{cases} |g_i^m| & \text{if } 0 < \alpha_i^m < C, \\ \max(0, -g_i^m) & \text{if } \alpha_i^m = 0, \\ \max(0, g_i^m) & \text{if } \alpha_i^m = C. \end{cases} \quad (18)$$

Clearly $v_i^m \geq 0$. Optimality holds when:

$$v_i^m = 0\ \forall m \neq i,\ \ \forall i. \quad (19)$$

For practical termination we can approximately check this using a tolerance parameter, $\epsilon > 0$:

$$v_i^m < \epsilon\ \forall m \neq i,\ \ \forall i. \quad (20)$$

The value $\epsilon = 0.1$ is a good choice for a practical implementation.

Like in section 2, the Sequential Dual Method (SDM) for the Weston-Watkins formulation consists of sequentially picking one i at a time and solving the restricted problem of optimizing only $\alpha_i^m\ \forall m \neq y_i$. To do this, we let $\delta\alpha_i^m$ denote the additive change to be applied to the current α_i^m, and optimize $\delta\alpha_i^m\ \forall m \neq y_i$. Let $\boldsymbol{\alpha}_i$, $\boldsymbol{\delta\alpha}_i$ and $\boldsymbol{g}_i$ be vectors that respectively gather the elements α_i^m, $\delta\alpha_i^m$ and g_i^m over all $m \neq y_i$. With $A_i = \|\boldsymbol{x}_i\|^2$ the sub-problem of optimizing $\boldsymbol{\delta\alpha}_i$ is given by

$$\begin{aligned} \min_{\boldsymbol{\delta\alpha}_i} \quad & \tfrac{1}{2}A_i\|\boldsymbol{\delta\alpha}_i\|^2 + \tfrac{1}{2}A_i(\mathbf{1}^T\boldsymbol{\delta\alpha}_i)^2 + \boldsymbol{g}_i^T\boldsymbol{\delta\alpha}_i \\ \text{s.t.} \quad & 0 \leq \boldsymbol{\delta\alpha}_i \leq C\mathbf{1} - \boldsymbol{\alpha}_i \end{aligned} \quad (21)$$

Like (10), the sub-problem (21) too has a simple form. We solve it also using the active set method. We note that the Hessian of the objective function in (21) is $A_i(\boldsymbol{I} + \mathbf{1}\mathbf{1}^T)$ (where $\boldsymbol{I}$ is the identity matrix), and so, any of its block diagonal sub-matrices can be analytically and cheaply inverted. Therefore the active-set method is very efficient and we used this method in our implementation. A complete description of SDM for the Weston-Watkins formulation is given in Algorithm 3.1. The initialization, as well as time and memory complexity analysis given earlier for Crammer-Singer SDM also hold for this algorithm. Turning to convergence, let us assume, like in section 2, that $A_i > 0\ \forall i$. Unlike (4), the dual (16) does not have any equality constraints. Algorithm 3.1 is a block coordinate descent method for which convergence results from the optimization literature (see [18] and in particular section 6 there) can be applied to show that Algorithm 3.1 has linear convergence. Finally, similar to section 2.2 we can argue that, the method of Hsu and Lin [10] is suited for nonlinear kernels, but SDM is better for the linear SVM case.

For efficiency, (21) can be solved only for some restricted variables, say only the $\delta\alpha_i^m$ for which $v_i^m > 0$. The heuristics given for Crammer-Singer SDM can be extended to the Weston-Watkins version too. In the shrinking phase, for each i, among all $m \neq i$ we only compute g_i^m and update

Algorithm 3.1 SDM for Weston-Watkins Formulation

- Initialize $\boldsymbol{\alpha}$ and the corresponding $\boldsymbol{w}_m\ \forall m$.
- Until (19) holds in an entire loop over examples do:

  For $i = 1, \ldots, l$

  (a) Compute $g_i^m\ \forall m \neq y_i$ and obtain v_i

  (b) If $\max_m v_i^m \neq 0$, solve (21) and set:

  $\alpha_i^m \leftarrow \alpha_i^m + \delta\alpha_i^m\ \ \forall m \neq y_i$

  Set $\delta\alpha_i^{y_i} = -\sum_{m\neq y_i} \delta\alpha_i^m$.

  $\boldsymbol{w}_m \leftarrow \boldsymbol{w}_m - \delta\alpha_i^m \boldsymbol{x}_i\ \forall m$

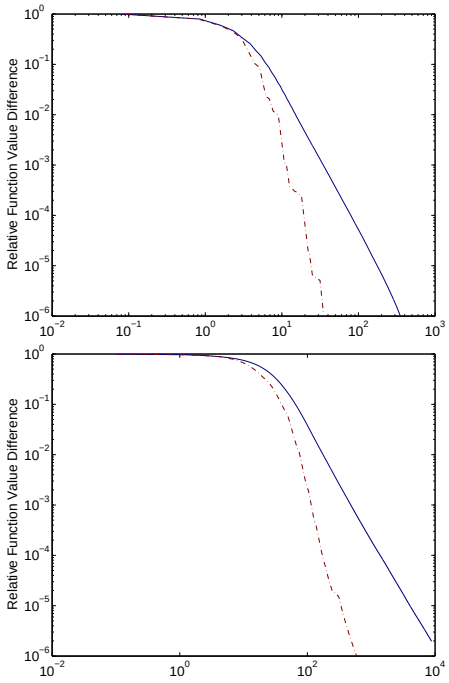

Figure 4: Comparison of *SDM* (Weston-Watkins) with (Red, dashdot) and without (Blue, solid) the shrinking and cooling heuristics on MNIST (top) and COVER (bottom) datasets. The horizontal axis is Training time in seconds.

α_i^m for those m which satisfy $0 < \alpha_i^m < C$. Figure 4 shows the effectiveness of including the heuristics on MNIST and COVER datasets.

4. COMPARISON WITH ONE-VERSUS-REST

The One-Versus-Rest (OVR) method which consists of developing, for each class m, a binary classifier ($\boldsymbol{w}_m$) that separates class m from the rest of the classes, and then using (2) for inference, is one of the popular schemes to solve a multi-class problem. For nonlinear SVMs, Rifkin and Klatau [21] argue that OVR is as good as any other approach if the binary classifiers are well-tuned regularized classifiers like SVMs. The main points in defense of OVR (and against direct multi-class methods) are: (1) complicated implementation of direct multi-class methods, (2) their slow training, and (3) OVR yields accuracy similar to that given by direct multi-class methods. These points should be viewed differently when considering linear SVMs. Firstly, a method such as SDM is easy to implement. Secondly, with respect to training speed, the following simplistic analysis gives insight. It is reasonable to take the cost of solution of a dual problem (whether it is a binary or a direct multi-class one) to be a function $T(l_d)$ where l_d is the number of dual variables in that problem.[§] In OVR we solve k binary problems each of which has l dual variables and so its cost is $kT(l)$. For a direct multi-class solution, e.g. Crammer-Singer, the number of dual variables is kl and therefore its cost is $T(kl)$. For nonlinear SVMs $T(\cdot)$ is a nonlinear function (some empirical studies have shown it to be close to a quadratic) and so OVR solution is much faster than direct multi-class solutions. On the other hand, for linear SVMs, $T(\cdot)$ tends to be a linear function and so the speeds of OVR and direct multi-class solutions are close. We observe this in an experiment given below. Thirdly, performance differences between OVR and direct multi-class methods is dependent on the dataset. The experiments below offer more insight on this aspect.

First we do an implementation of OVR in which each SVM binary classifier employing the standard hinge loss is solved using the binary version of SDM, as developed in [9]. By the results in [9] this is probably the fastest solver for OVR for the types of datasets considered in this paper. We begin with a simple experiment where we fix $C = 1$ and solve the multi-class problems using OVR and SDM for Crammer-Singer and Weston-Watkins formulations. For MNIST and COVER datasets Figure 5 gives the behavior of test set performance as the solutions progress in time. For the OVR method test set performance was evaluated after each loop of the training set in each binary classifier solution, whereas, for the direct multi-class SDM the evaluation was done on a finer scale, four times within one loop. Clearly, all three methods take nearly the same amount of computing time to reach their respective steady state values. This is in agreement with our earlier mention that, for linear SVMs OVR and direct multi-class solutions have similar complexity. Also, in these datasets the final test set accuracies of Crammer-Singer and Weston-Watkins formulations are better than the accuracy of OVR.

Since accuracy depends on the choice of C, we do a more detailed study on the accuracy aspects further. For each of the 5 datasets, we combined the training and test datasets and make 10 random 60%/20%/20% train/validation/test class-stratified partitions. For each of the three multi-class methods the C parameter is tuned using the train and validation sets by considering the following set of C values: $\{0.01, 0.03, 0.1, 0.3, 1, 3, 10\}$. For the best C value that gives the largest validation set accuracy, a re-training is done on the train+validation set and the resulting classifier is evaluated on the test set. The mean and standard deviation values of test set accuracies are reported in Table 4.2. We conduct Wilcoxon sign-rank test with a significance level of 0.01 to compare OVR against the direct multi-class methods on each dataset; see Table 4.3. Comparison of OVR and Crammer-Singer indicates that the observed differences are statistically significant in favor of Crammer-Singer for SECTOR, MNIST, RCV1 and COVER datasets. Comparison of OVR and Weston-Watkins indicates that the results are statistically significant for all the datasets. However, on NEWS20 and SECTOR the results are in favor of OVR while on the other three datasets they are in favor of Weston-Watkins. Though statistical significance analysis may not mean practically significant improvements, we do note that the improvements shown by the direct multi-class methods over OVR on MNIST and COVER datasets are quite good. Overall, for linear SVMs it may be better to employ the direct multi-class methods (in particular, Crammer-Singer) whenever fast solvers for them (such as ones based on SDM) are available. It is worth noting that, of the five datasets under consideration only MNIST and COVER are un-normalized, i.e. $\|\boldsymbol{x}_i\|$ is not the same for all i. It is possible that OVR's performance is more affected by

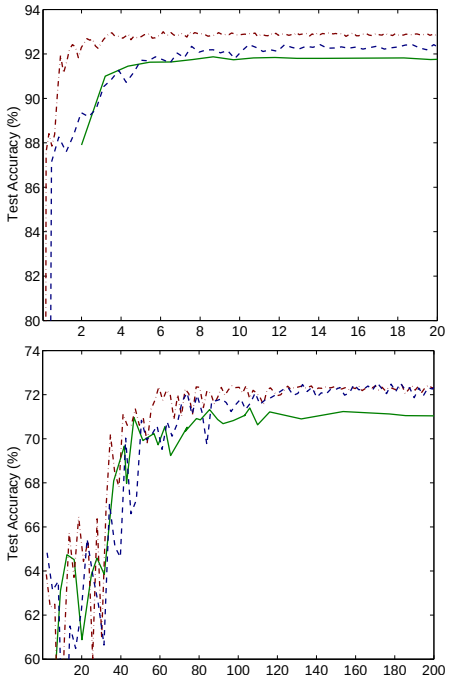

Figure 5: Comparison of One-Versus-Rest (Green, solid), Crammer-Singer (Red, dashdot) and Weston-Watkins (Blue, dashed) on MNIST (top) and COVER (bottom) datasets. The horizontal axis is Training time in seconds.

[§]This can be seen from the complexity analysis of section 2 coupled with the assumption that a fixed small number of loops gives good enough convergence.

Table 4.2: Test set accuracy (Mean and Standard deviation) comparison of One-Versus-Rest (OVR), and Multi-class SDMs for Crammer-Singer (CS) and Weston-Watkins (WW) formulations with C tuned for each method.

Dataset	OVR	CS	WW
NEWS20	85.39 ± 0.37	85.26 ± 0.37	85.10 ± 0.43
SECTOR	94.83 ± 0.56	95.17 ± 0.62	94.37 ± 0.57
MNIST	91.46 ± 0.23	92.50 ± 0.20	91.84 ± 0.21
RCV1	90.85 ± 0.10	91.19 ± 0.07	91.10 ± 0.11
COVER	71.35 ± 0.33	72.31 ± 0.37	72.40 ± 0.25

Table 4.3: p-values from Wilcoxon sign-rank test.

Dataset	OVR-CS	OVR-WW
NEWS20	0.43	0.01
SECTOR	0.01	3.9×10^{-3}
MNIST	2.0×10^{-3}	2.0×10^{-3}
RCV1	2.0×10^{-3}	2.0×10^{-3}
COVER	2.0×10^{-3}	2.0×10^{-3}

the lack of data normalization and therefore direct multi-class solutions may be more preferred in such situations. A more detailed analysis is needed to check this.

5. CONCLUSION

In this paper we have presented sequential descent methods for the Crammer-Singer and the Weston-Watkins multi-class linear SVM formulations that are well suited for solving large scale problems. The basic idea of sequentially looking at one example at a time and optimizing the dual variables associated with it can be extended to more general SVM problems with structured outputs, such as taxonomy, multi-label and sequence labeling problems. We note that, for the structured outputs setting, Tsochantaridis et al. [24] mentioned the sequential dual method as a possibility (see Variant (b) in Algorithm 1 of that paper), but did not pursue it as a potentially fast method. In such extensions the quadratic programming sub-problem associated with each example does not have a simple form such as those in (10) and (21), and some care is needed to solve this sub-problem efficiently. We are currently conducting experiments on such extensions and will report results in a future paper.

6. REFERENCES

[1] A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with LaRank. In *ICML*, 2007.

[2] B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, 1992.

[3] L. Bottou. Stochastic gradient descent examples, 2007. `http://leon.bottou.org/projects/sgd`.

[4] M. Collins, A. Globerson, T. Koo, X. Carreras, and P. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks. *JMLR*, 2008. To appear.

[5] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *COLT*, 2000.

[6] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, 2:265–292, 2001.

[7] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *JMLR*, 3:951–991, 2003.

[8] R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, 1987.

[9] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *ICML*, 2008.

[10] C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE TNN*, 13(2):415–425, 2002.

[11] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.

[12] T. Joachims. Training linear SVMs in linear time. In *ACM KDD*, 2006.

[13] T. Joachims, T. Finley, and C.-N. J. Yu. Cutting plane training of structural SVMs. *MLJ*, 2008.

[14] K. Lang. Newsweeder: Learning to filter netnews. In *ICML*, pages 331–339, 1995.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. MNIST database available at `http://yann.lecun.com/exdb/mnist/`.

[16] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397, 2004.

[17] C.-J. Lin. A formal analysis of stopping criteria of decomposition methods for support vector machines. *IEEE TNN*, 13(5):1045–1052, 2002.

[18] Z.-Q. Luo and P. Tseng. On the convergence of coordinate descent method for convex differentiable minimization. *JOTA*, 72(1):7–35, 1992.

[19] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *Proceedings of the AAAI'98 Workshop on Learning for Text categorization*, 1998.

[20] J. D. M. Rennie and R. Rifkin. Improving multiclass text classification with the Support Vector Machine. Technical Report AIM-2001-026, MIT, 2001.

[21] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *JMLR*, 5:101–141, 2004.

[22] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: primal estimated sub-gradient solver for SVM. In *ICML*, 2007.

[23] C. H. Teo, A. Smola, S. V. Vishwanathan, and Q. V. Le. A scalable modular convex solver for regularized risk minimization. In *ACM KDD*, 2007.

[24] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 2005.

[25] J. Weston and C. Watkins. Multi-class support vector machines. In M. Verleysen, editor, *Proceedings of ESANN99*, Brussels, 1999. D. Facto Press.

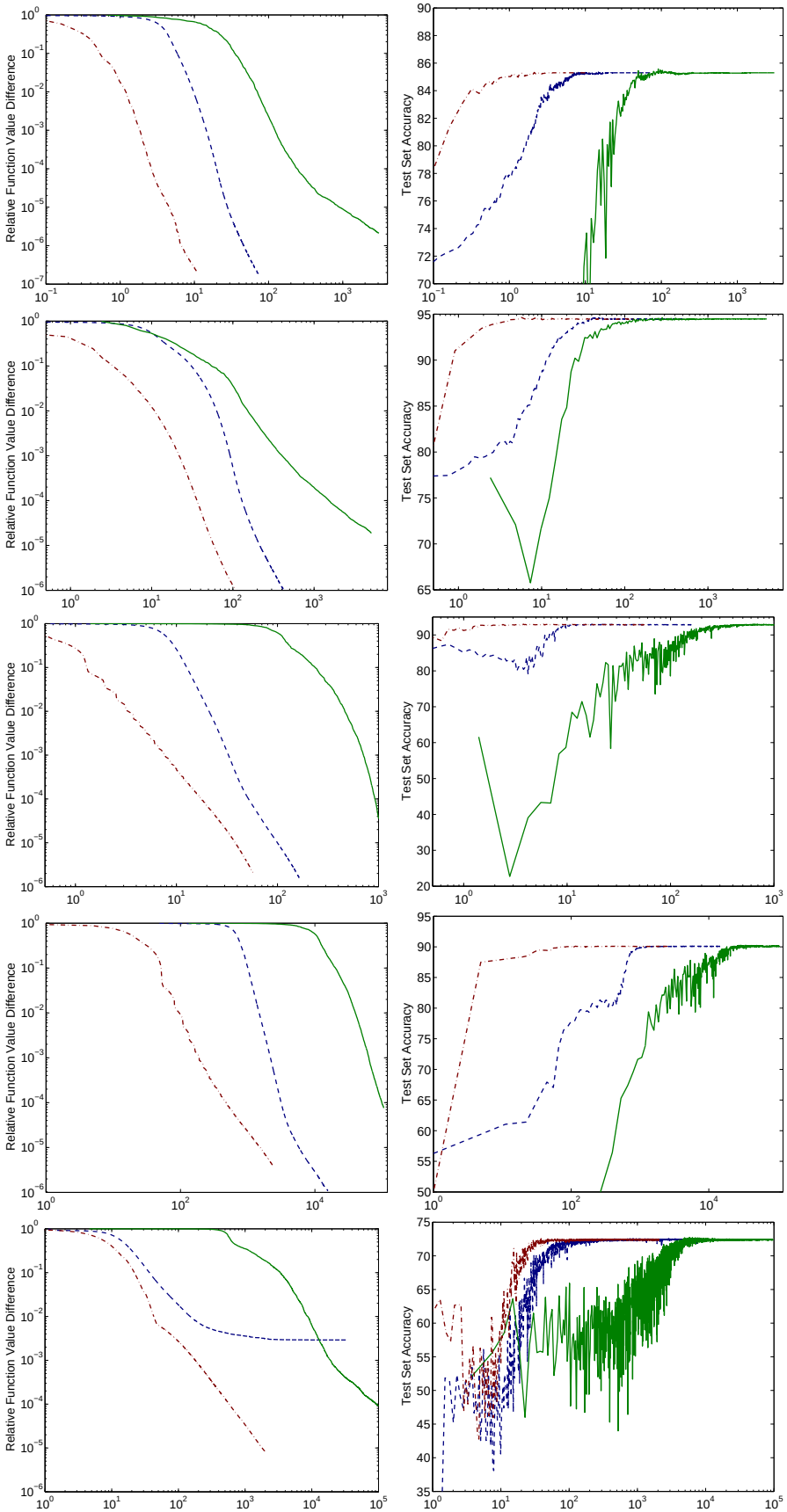

Figure 3: Comparison of *SDM* (Red, dashdot), *EG* (Blue, dashed) and *Bundle* (Green, solid) on various datasets with C=1: The horizontal axis is Training time in seconds. Each row corresponds to one dataset; Row 1: NEWS20; Row 2: SECTOR; Row 3: MNIST; Row 4: RCV1; and Row 5: COVER.