Large-scale Multiclass Support Vector Machine Training via Euclidean Projection onto the Simplex

Mathieu Blondel, Akinori Fujino, Naonori Ueda
NTT Communication Science Laboratories
Kyoto, Japan

***Abstract*—Dual decomposition methods are the current state-of-the-art for training multiclass formulations of Support Vector Machines (SVMs). At every iteration, dual decomposition methods update a small subset of dual variables by solving a restricted optimization problem. In this paper, we propose an *exact* and *efficient* method for solving the restricted problem. In our method, the restricted problem is reduced to the well-known problem of Euclidean projection onto the positive simplex, which we can solve exactly in expected $O(k)$ time, where k is the number of classes. We demonstrate that our method empirically achieves state-of-the-art convergence on several large-scale high-dimensional datasets.**

I. Introduction

Support Vector Machines (SVMs) [1] are arguably one of the most popular algorithms for classification. Although SVMs were originally designed for binary classification, many approaches have since then been proposed to tackle multiclass classification. Roughly, these approaches can be divided into two categories. On one hand, indirect approaches such as one-vs-rest (a.k.a. one-vs-all), one-vs-one and error-correcting output codes reduce multiclass classification to multiple binary classification problems. On the other hand, multiclass SVM formulations [2]–[4] tackle multiclass classification directly by minimizing a single objective function. In this paper, we focus on the Crammer-Singer formulation [3], which is by far the most popular direct multiclass SVM formulation.

When it comes to training multiclass SVMs, dual decomposition (a.k.a. dual block coordinate descent/ascent) methods have been shown [5], [6] to outperform other methods such as exponentiated gradient [7], cutting-plane [8] and stochastic (sub)-gradient [9]. At every iteration, dual decomposition methods update a small subset of dual variables by solving a restricted optimization problem. In their original paper, Crammer and Singer [3] proposed to solve this problem numerically using a fixed point method. Keerthi et al. [5] studied the Crammer-Singer formulation with the linear kernel in mind and proposed to solve the problem using an active set method for quadratic programming. In the structured SVM literature, Bordes et al. [10], as well as Balamurugan et al. [6], used Sequential Minimal Optimization (SMO) [11] to solve the restricted optimization problem. More recently, Lacoste-Julien et al. [12] proposed a block Frank-Wolfe method, which solves a linear approximation of the restricted problem.

In this paper, we propose an *exact* and *efficient* method for solving the restricted problem. In our method, the restricted problem is reduced to the well-known problem of Euclidean projection onto the positive simplex, which we can solve exactly in expected $O(k)$ time, where k is the number of classes. We conduct an extensive empirical comparison of several methods for solving the restricted problem and demonstrate on several large-scale high-dimensional datasets that our exact method for solving the restricted problem is computationally cheap and leads to fast convergence in practice.

II. Multiclass SVMs

Multiclass SVMs classify an input vector $\boldsymbol{x} \in \mathbf{R}^d$ into one of k classes using the following simple rule:

$$\hat{y} = \operatorname*{argmax}_{m \in [k]} \boldsymbol{w}_m^{\mathrm{T}} \boldsymbol{x}. \tag{1}$$

Each vector $\boldsymbol{w}_m \in \mathbf{R}^d$ can be thought as a prototype representing the m^{th} class and the inner product $\boldsymbol{w}_m^{\mathrm{T}} \boldsymbol{x}$ as the score of the m^{th} class with respect to $\boldsymbol{x}$. Therefore, Eq. (1) chooses the class with highest score. Given n training instances $\boldsymbol{x}_i \in \mathbf{R}^d$ and their associated labels $y_i \in [k]$, the Crammer-Singer multiclass SVM formulation [3] estimates $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k$ by solving the following optimization problem:

$$\underset{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_k}{\text{minimize}} \; \frac{1}{2} \sum_{m=1}^{k} \|\boldsymbol{w}_m\|^2 + C \sum_{i=1}^{n} \left[1 + \max_{m \neq y_i} \boldsymbol{w}_m^{\mathrm{T}} \boldsymbol{x}_i - \boldsymbol{w}_{y_i}^{\mathrm{T}} \boldsymbol{x}_i \right]_+, \tag{2}$$

where $C > 0$ is a regularization parameter and $[u]_+ = 0$ if $u < 0$ and u otherwise. Intuitively, Eq. (2) means that, for each training instance, we suffer no loss if the score of the correct class is larger than the score of the "closest" class by at least 1. In the remainder of this paper, we assume that $\|\boldsymbol{x}_i\| > 0$, since any $\boldsymbol{x}_i$ with $\|\boldsymbol{x}_i\| = 0$ (which can only happen if $\boldsymbol{x}_i = \boldsymbol{0}$) does not affect the solution of Eq. (2). The dual of Eq. (2) is given by [3], [5]

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{minimize}} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2} \sum_{m=1}^{k} \|\boldsymbol{w}_m(\boldsymbol{\alpha})\|^2 + \sum_{i=1}^{n} \sum_{m=1}^{k} \Delta_i^m \alpha_i^m \\ \text{subject to} \quad & \alpha_i^m \leq C_i^m \quad \forall i \in [n] \quad \forall m \in [k] \\ & \sum_{m=1}^{k} \alpha_i^m = 0 \quad \forall i \in [n], \end{aligned} \tag{3}$$

where $C_i^m = C$ if $m = y_i$ and $C_i^m = 0$ otherwise, $\Delta_i^m = 0$ if $m = y_i$ and 1 otherwise . The primal-dual relationship is given by

$$\boldsymbol{w}_m(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i^m \boldsymbol{x}_i \quad \forall m \in [k].$$

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

$$g_i^m = \frac{\partial f}{\partial \alpha_i^m} = \boldsymbol{w}_m(\boldsymbol{\alpha})^{\mathrm{T}} \boldsymbol{x}_i + \Delta_i^m \quad \forall i \in [n] \; \forall m \in [k]. \tag{4}$$

Following [3], we have an optimal solution if and only if $v_i = 0 \ \forall i \in [n]$, where

$$v_i = \max_{m \in [k]} g_i^m - \min_{m \in [k]: \alpha_i^m < C_i^m} g_i^m \quad \forall i \in [n]. \tag{5}$$

The larger v_i, the more $\alpha_i^1, \ldots, \alpha_i^k$ violate the optimality conditions. In practice, given a tolerance parameter ϵ, we can stop an optimization algorithm if $v_i < \epsilon \quad \forall i \in [n]$.

III. Dual decomposition for multiclass SVMs

The main idea of dual decomposition methods is to update at every iteration a small subset of dual variables, keeping all others fixed. Since the variables $\alpha_i^1, \ldots, \alpha_i^k$ associated with $\boldsymbol{x}_i$ are tied by an equality constraint, a natural choice for decomposition methods is to update these variables as a block. Let $\boldsymbol{\alpha}_i$, $\boldsymbol{g}_i$ and $\boldsymbol{C}_i$ be k-dimensional vectors that gather elements α_i^m, g_i^m and C_i^m for all $m \in [k]$. Our goal is to find $\boldsymbol{\delta}_i \in \mathbf{R}^k$ such that the update $\boldsymbol{\alpha}_i \leftarrow \boldsymbol{\alpha}_i + \boldsymbol{\delta}_i$ maximizes the decrease of the dual objective. Then the restricted problem is

$$\begin{aligned} \underset{\boldsymbol{\delta}_i}{\text{minimize}} \quad & \hat{f}(\boldsymbol{\delta}_i) = \frac{\|\boldsymbol{x}_i\|^2}{2} ||\boldsymbol{\delta}_i||^2 + \boldsymbol{g}_i^{\mathrm{T}} \boldsymbol{\delta}_i \\ \text{subject to} \quad & \boldsymbol{\delta}_i \leq \boldsymbol{C}_i - \boldsymbol{\alpha}_i \\ & \boldsymbol{\delta}_i^{\mathrm{T}} \mathbf{1} = 0, \end{aligned} \tag{6}$$

where $\mathbf{1}$ is the k-dimensional vector of all ones and $\boldsymbol{\alpha}_i$ is fixed to its current estimate. This can be derived by using the second-order Taylor expansion $\hat{f}(\boldsymbol{\delta}_i) = \boldsymbol{g}_i^{\mathrm{T}} \boldsymbol{\delta}_i + \frac{1}{2} \boldsymbol{\delta}_i^{\mathrm{T}} \boldsymbol{H}_i \boldsymbol{\delta}_i$, where $\boldsymbol{H}_i \in \mathbf{R}^{k \times k}$ is the partial Hessian matrix associated with $\boldsymbol{x}_i$. It is easy to see that $\boldsymbol{H}_i = \|\boldsymbol{x}_i\|^2 \boldsymbol{I}$ (i.e., a diagonal matrix), thus leading to the simple form in Eq. (6). A non-linear kernel κ can easily be used by replacing inner products $\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$ in Eq. (4) by $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\|\boldsymbol{x}_i\|^2$ in Eq. (6) by $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_i)$.

We summarize dual decomposition for multiclass SVMs assuming a linear kernel in Algorithm 1. Note that we update the vectors $\boldsymbol{\alpha}_i$ in cyclic order after shuffling the data. Cyclic or random selection are known to be the most cost-effective schemes for the linear kernel [5]. For non-linear kernels, gradient-based selection schemes are usually used [3]. Decomposition methods were shown to converge to a global solution in a finite number of steps in [13].

Solving Eq. (6) is the core problem that must be solved in order to train multiclass SVMs by dual decomposition. It is possible to solve Eq. (6) using a generic quadratic program solver. However, that would take $O(k^2)$ time. Therefore, several works in the literature have investigated developing efficient algorithms. Crammer and Singer originally proposed two methods for solving Eq. (6): an exact procedure based on sorting with complexity $O(k \log k)$ [14] and an approximate fixed-point method with complexity $O(k)$ [3]. Keerthi et al. [5] used an active set method for quadratic programming, unfortunately, omitting the details. The problem in Eq. (6) or related problems were also investigated in the structured SVM literature. Bordes et al. [10], as well as Balamurugan et al. [6], used Sequential Minimal Optimization (SMO) [11] to obtain an approximate solution. More recently, Lacoste-Julien et al. [12] proposed a block Frank-Wolfe method. The main idea of their method is to solve a *linear* approximation of the restricted problem (i.e., without Hessian term), for which there exists a closed-form solution.

Algorithm 1 Dual decomposition for multiclass SVMs

Input: $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, $C > 0$, $\epsilon > 0$
Initialize $\boldsymbol{\alpha} \leftarrow \mathbf{0}$ and $\boldsymbol{w}_m \leftarrow \mathbf{0} \ \forall m \in [k]$
Shuffle data
repeat
  $v_{max} \leftarrow -\infty$
  for $i \in [n]$ **do**
    Compute violation v_i by Eq. (5)
    $v_{max} \leftarrow \max(v_{max}, v_i)$
    if $v_i > 0$ **then**
      Find $\boldsymbol{\delta}_i$ by solving Eq. (6)
      $\boldsymbol{\alpha}_i \leftarrow \boldsymbol{\alpha}_i + \boldsymbol{\delta}_i$ and $\boldsymbol{w}_m \leftarrow \boldsymbol{w}_m + \delta_i^m \boldsymbol{x}_i \ \forall m \in [k]$
    end if
  end for
until $v_{max} \leq \epsilon$
Output: $\boldsymbol{\alpha}$ and $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k$

IV. Reduction to Projection onto the Simplex

While upon first inspection the optimization problem in Eq. (6) looks very specific to the dual problem in Eq. (3), we show in this section that we can reduce the problem to a much more standard form. Lemma 1 characterizes our reduction.

Lemma 1: Let $\hat{\boldsymbol{\beta}} = ||\boldsymbol{x}_i||(\boldsymbol{C}_i - \boldsymbol{\alpha}_i) + \frac{\boldsymbol{g}_i}{||\boldsymbol{x}_i||}$ and $z = C||\boldsymbol{x}_i||$. If $\boldsymbol{\beta}$ is the optimal solution of

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{minimize}} \quad & \frac{1}{2} ||\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}||^2 \\ \text{subject to} \quad & \boldsymbol{\beta} \geq 0 \\ & \boldsymbol{\beta}^{\mathrm{T}} \mathbf{1} = z, \end{aligned} \tag{7}$$

then the optimal solution of Eq. (6) is $\boldsymbol{\delta}_i = \boldsymbol{C}_i - \boldsymbol{\alpha}_i - \frac{\boldsymbol{\beta}}{||\boldsymbol{x}_i||}$.

The problem in Eq. (7) is known as the *Euclidean projection onto the positive simplex* when $z > 0$ and onto the *probabilistic simplex* when $z = 1$ [15]. Geometrically, the problem corresponds to finding the closest point to $\hat{\boldsymbol{\beta}}$ which is both in the non-negative orthant and on the hyperplane defined by $\boldsymbol{\beta}^{\mathrm{T}} \mathbf{1} = z$.

While previous studies solved Eq. (6), Lemma 1 shows that we can solve Eq. (7) instead. In other words, we can use the Euclidean projection onto the positive simplex as a "building block" for training multiclass SVMs. This is very useful because the projection is a standard problem which appears in many fields, notably in imaging and statistics. It is also known that the projection onto the ℓ_1-ball can be reduced to the projection onto the positive simplex [15]. Furthermore, as we present in the sequel, the projection on the positive simplex can be computed exactly in expected $O(k)$ time. No algorithm with this type of guarantee was reported in the SVM literature for solving Eq. (6).

We now briefly present how to compute the projection onto the positive simplex. The Lagrangian of the problem in Eq. (7) is

$$\mathcal{L} = \frac{1}{2} \|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|^2 + \theta \Big(\sum_{m=1}^k \beta_m - z \Big) - \boldsymbol{\omega}^{\mathrm{T}} \boldsymbol{\beta},$$

where $\theta \in \mathbf{R}$ is Lagrange multiplier and $\boldsymbol{\omega} \in \mathbf{R}_+^k$ is a vector of non-negative Lagrange multipliers. Differentiating

the Lagrangian with respect to β_m and solving for zero gives the optimality condition

$$\beta_m = \hat{\beta}_m - \theta + \omega_m.$$

The KKT complementary slackness conditions require that $\beta_m \omega_m = 0 \ \forall m$. It follows that if $\beta_m > 0$, then $\omega_m = 0$ and thus $\beta_m = \hat{\beta}_m - \theta$. Otherwise, the non-negativity constraint implies that $\beta_m = 0$. Wrapping up the two cases, the solution must take the following form:

$$\beta_m = \left[\hat{\beta}_m - \theta\right]_+.$$

Intuitively, $\beta_1, \ldots, \beta_k$ are tied by a single variable $\theta \in \mathbf{R}$. Therefore, finding $\boldsymbol{\beta} \in \mathbf{R}^k$ reduces to finding $\theta \in \mathbf{R}$. We present two methods for finding θ.

The first method, due to Shalev-Shwartz and Singer [16], is based on sorting. The authors show that if we sort $\hat{\boldsymbol{\beta}}$ into $\boldsymbol{\mu}$ and define the function $\pi(m) \equiv \frac{1}{m}\left(\sum_{r=1}^{m} \mu_r - z\right)$, then the *exact* solution is $\theta = \pi(\rho)$, where $\rho = \max\left\{m \in [k] : \mu_m - \pi(m) > 0\right\}$. The procedure, which takes $O(k \log k)$ because of sorting, is summarized below.

Algorithm 2 Sort algorithm for projection onto the simplex

Input: $\hat{\boldsymbol{\beta}} \in \mathbf{R}^k$, a scalar $z > 0$
Sort $\hat{\boldsymbol{\beta}}$ into $\boldsymbol{\mu}$: $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_k$
Find $\rho = \max\left\{m \in [k] : \mu_m - \pi(m) > 0\right\}$
Output: $\boldsymbol{\beta}$ where $\beta_m = \left[\hat{\beta}_m - \pi(\rho)\right]_+ \quad \forall m \in [k]$

This complexity can be reduced to expected $O(k)$ time while maintaining exactness of the solution by using a randomized pivot algorithm [15]. The pivot algorithm does not need to sort $\hat{\boldsymbol{\beta}}$ beforehand: informally, it computes the solution "on the fly" while partially sorting $\hat{\boldsymbol{\beta}}$. See Figure 2 of [15] for the algorithm.

The sort and pivot algorithms compute θ *exactly* in $O(k \log k)$ and *expected* $O(k)$ time, respectively. We now present a method for computing θ *approximately* in *worst-case* $O(k)$ time. The method is based on casting the projection onto the simplex as a root finding problem. Lemma 2 characterizes the root finding problem and gives some useful properties.

Lemma 2: The optimal solution of Eq. (7) is $\boldsymbol{\beta}$ where $\beta_m = \left[\hat{\beta}_m - \theta\right]_+ \ \forall m \in [k]$ and θ is the root of $\phi(t) = \sum_{m=1}^{k} \left[\hat{\beta}_m - t\right]_+ - z$. Moreover, the following properties hold:

1) $\hat{\boldsymbol{\beta}}^{\mathrm{T}} \mathbf{1} > z$ if $v_i > 0$
2) $\phi(t)$ is a continuous decreasing function on $[0, \infty)$
3) $\phi(0) > 0$
4) $\phi(U) < 0$ where $U = \max_m \hat{\beta}_m$

The proof is similar to the proof of Theorem 1 in [17] and of Lemma 2 in [18]. Property 1 of Lemma 2 is specific to our reduction and geometrically means that $\hat{\boldsymbol{\beta}}$ is always above the hyperplane defined by $\boldsymbol{\beta}^{\mathrm{T}} \mathbf{1} = z$. Thus, to satisfy $\boldsymbol{\beta}^{\mathrm{T}} \mathbf{1} = z$, we must necessarily have $\theta > 0$. Properties 2 to 4 imply that $\phi(t)$ must necessarily cross zero between 0 and U (Intermediate Value Theorem) and thus $\theta \in (0, U]$. While any root finding suitable for non-differentiable functions can be used, we recommend to use bisection, which is both conceptually simple and finds an approximate solution in worst-case $O(k)$ complexity. Starting from $[0, U]$, bisection works by repeatedly halving the current interval and selecting the subinterval in which the root must lie. We summarize the procedure below.

Algorithm 3 Bisection for projection onto the simplex

Input: $\hat{\boldsymbol{\beta}} \in \mathbf{R}^k$, a scalar $z > 0$, a tolerance parameter τ
$l \leftarrow 0 \quad u \leftarrow U \quad s \leftarrow \infty$
while $|s|/z > \tau$ **do**
  $\theta \leftarrow (l + u)/2$
  $s \leftarrow \phi(\theta)$
  $u \leftarrow \theta$ if $s < 0$ or $l \leftarrow \theta$ otherwise
end while
Output: $\boldsymbol{\beta}$ where $\beta_m = \left[\hat{\beta}_m - \theta\right]_+ \quad \forall m \in [k]$

Before solving Eq. (7) by Algorithm 2 or 3, we need to compute $\hat{\boldsymbol{\beta}}$, which depends on the partial gradient $\boldsymbol{g}_i$. For the linear kernel, computing $\boldsymbol{g}_i$ takes $O(k\bar{d})$ time, where $\bar{d}$ is the average number of non-zero features per training instance. This is larger than $O(k)$ and typically larger than $O(k \log k)$ as well. For non-linear kernels, the cost of computing $\boldsymbol{g}_i$ is even larger, $O(nk\bar{d})$. Therefore, the cost of computing $\boldsymbol{g}_i$ outweighs the cost of solving Eq. (7), even exactly. Our experiments confirm that solving Eq. (7) exactly is computationally cheap and leads to fast convergence in practice.

In [7] and [12], the exponentiated gradient (EG) and Frank-Wolfe (FW) methods were used to solve the dual objective of structured SVMs, which is quadratic function with simplex constraints. However, EG and FW solve the restricted problem only approximately (FW solves a linear approximation of the restricted problem, i.e., without Hessian term). Furthermore, the restricted problem for the dual in [7], [12] cannot be reduced to the same form as Eq. (7).

V. Empirical evaluation

We conducted experiments on 4 publicly available high-dimensional datasets: Amazon7 [19], [20] (product reviews), RCV1 (news documents), News20 (newsgroup messages) and Sector (web-pages). The dataset characteristics are summarized in Table I. All results reported in this section were obtained by repeating each experiment 10 times with a different train/test split. Each time, we used stratified selection in order to split the dataset into $4/5$ training and $1/5$ testing. Unless otherwise specified, throughout our experiments we used $C = 1$ for RCV1, News20, Sector, and $C = 0.01$ for Amazon7. In all experiments, we used the linear kernel.

TABLE I: Datasets used in this section.

Dataset	Instances	Features	Non-zero features	Classes
Amazon7	1,362,109	262,144	0.04%	7
RCV1	534,135	47,236	0.1%	52
News20	18,846	130,088	0.1%	20
Sector	9,619	55,197	0.3%	105

A. *Comparison between proposed and existing methods*

We compared the following methods for solving the restricted problem Eq. (6) or its equivalent form Eq. (7):

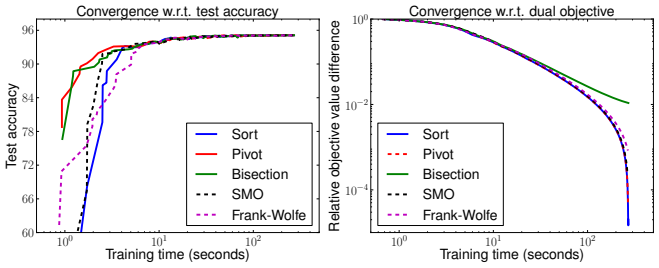
- Sort: solves Eq. (7) exactly in $O(k \log k)$ time (c.f., Algorithm 2)
- Pivot: solves Eq. (7) exactly in expected $O(k)$ time (c.f., Figure 2 of [15])
- Bisection: solves Eq. (7) approximately in $O(k)$ time (c.f., Algorithm 3)
- Sequential Minimal Optimization (SMO): solves Eq. (6) approximately by iteratively updating pairs of variables
- Frank-Wolfe (FW): solves a linear approx. of Eq. (6)

Details regarding the SMO and FW methods are given in the supplementary material. For bisection, we set the tolerance parameter of the stopping criterion to $\tau = 10^{-3}$. We investigate the effect of τ in Section V-B. Since, in our implementation, the above methods are all used within the same dual decomposition algorithm template (Algorithm 1), we can be sure that any difference between these methods is due to the way we solve the restricted problem and not due to other design choices or implementation details.

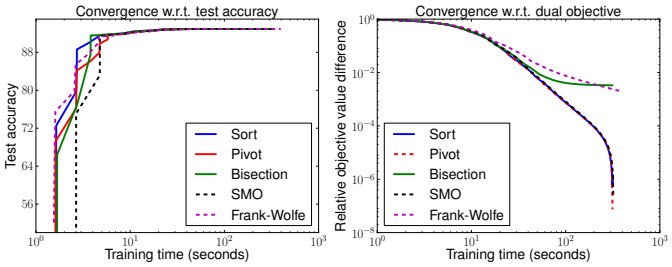
Convergence results. We compared the convergence of the above methods with respect to the dual objective value (lower is better) and test accuracy (higher is better). Results are given in Fig. 1. With respect to dual objective value, methods that solve the restricted problem exactly (Sort, Pivot) constantly achieved the fastest convergence. On the other hand, methods that solve the restricted problem approximately (Frank-Wolfe, Bisection) were found to converge slowly. This is not surprising since, as we discussed in Section IV, the cost of computing the partial gradient $\boldsymbol{g}_i$ outweighs the cost of solving the restricted problem. Therefore, it is worth solving the restricted problem exactly. With respect to test accuracy, we found that exact methods were the fastest on Amazon7/RCV1/News20 and comparable to approximate methods on Sector.

Training with only one or few passes. In a large-scale setting, one can usually only afford to make few passes over the training set, or sometimes even just one. In one-pass training, training instances can be vectorized on the fly (e.g., using the hashing trick [21]) and released from memory as soon as the model has been updated. Dual decomposition methods are ideal candidates in this setting, since they can be used online: at any given time, dual decomposition methods only require access to a single training instance $\boldsymbol{x}_i$. Table II indicates our results in such a low-computational budget setting: 1, 3, 5 passes over the training data. We found that methods that solve the restricted problem exactly (Sort, Pivot) constantly achieved the best accuracy. For example, on News20, Sort and Pivot achieved 83.51% and 83.85% accuracy in one pass, respectively, while SMO and Frank-Wolfe only achieved 82.55% and 81.23%. This is not surprising, since SMO and Frank-Wolfe solve the restricted problem only partially. Our experimental results therefore indicate that solving the restricted problem exactly is beneficial during the first few passes.

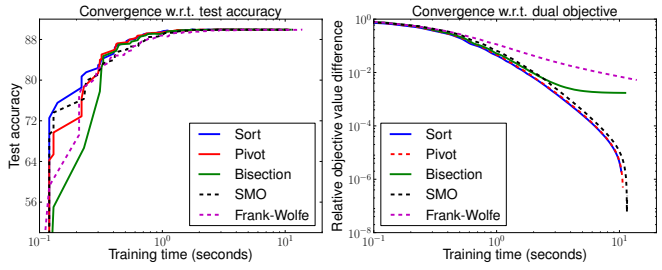
In terms of training time, our results confirm that solving the restricted problem exactly is not more computationally



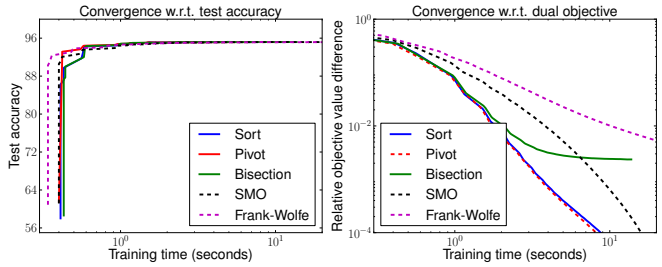
(a) Amazon7



(b) RCV1



(c) News20



(d) Sector

Fig. 1: Convergence of proposed and existing methods with respect to test accuracy and dual objective function. Relative objective value difference is computed by $|f(\boldsymbol{\alpha}) - f(\boldsymbol{\alpha}^*)|/|f(\boldsymbol{\alpha}^*)|$, where $\boldsymbol{\alpha}^*$ is obtained by solving the problem very accurately.

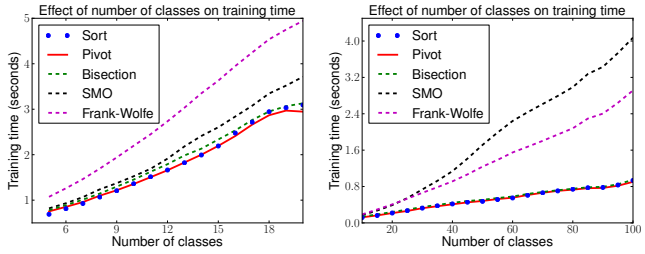
expensive than other methods. For example, on Amazon7, it took 5.98 and 6.01 seconds to Sort and Pivot for making one pass over the training data, while SMO and Frank-Wolfe took 6.06 and 5.97 seconds, respectively.

Scalability with respect to the number of classes. To investigate the effect of the number of classes k on training time, we used the News20 and Sector datasets and created samples containing a subset of the classes. The samples contained $5, 6, \ldots, 20$ classes for News20 and $10, 15 \ldots, 105$ classes for Sector. Since our goal was to measure the impact of the number of classes and not of the number of training instances, we used the same number of training instances in

TABLE II: Method comparison for 1, 3 and 5 passes over the training data. Time is in seconds. Accuracy is w.r.t. the test set.

Dataset	Passes		Sort	Pivot	Bisec	SMO	FW
Amazon7	1	Acc	**93.49 ± 0.51**	93.41 ± 0.78	92.95 ± 0.78	92.62 ± 1.07	93.19 ± 0.66
		Time	5.98 ± 0.02	6.01 ± 0.03	6.17 ± 0.02	6.06 ± 0.02	5.97 ± 0.02
	3	Acc	**94.49 ± 0.64**	94.47 ± 0.43	94.47 ± 0.46	94.29 ± 0.52	94.36 ± 0.51
		Time	17.13 ± 0.04	17.19 ± 0.07	17.77 ± 0.03	17.39 ± 0.03	16.94 ± 0.03
	5	Acc	94.81 ± 0.33	**94.87 ± 0.14**	94.78 ± 0.24	94.81 ± 0.44	94.85 ± 0.18
		Time	28.16 ± 0.08	28.30 ± 0.06	29.14 ± 0.07	28.62 ± 0.07	28.01 ± 0.04
RCV1	1	Acc	**91.91 ± 0.14**	91.90 ± 0.11	91.86 ± 0.09	91.80 ± 0.09	91.66 ± 0.10
		Time	8.96 ± 0.01	8.83 ± 0.02	9.15 ± 0.02	8.86 ± 0.03	8.71 ± 0.05
	3	Acc	92.82 ± 0.04	**92.84 ± 0.04**	92.82 ± 0.03	92.81 ± 0.03	92.74 ± 0.04
		Time	25.90 ± 0.03	25.73 ± 0.05	26.40 ± 0.06	25.75 ± 0.04	25.17 ± 0.07
	5	Acc	**92.90 ± 0.03**	**92.90 ± 0.03**	92.89 ± 0.04	92.89 ± 0.05	92.87 ± 0.03
		Time	42.99 ± 0.11	42.45 ± 0.08	43.40 ± 0.08	42.97 ± 0.15	41.63 ± 0.07
News20	1	Acc	83.51 ± 1.04	**83.85 ± 0.87**	83.10 ± 0.76	82.55 ± 1.38	81.23 ± 1.89
		Time	0.33 ± 0.03	0.31 ± 0.01	0.31 ± 0.01	0.31 ± 0.01	0.27 ± 0.01
	3	Acc	**89.20 ± 0.35**	89.12 ± 0.39	89.10 ± 0.47	88.87 ± 0.50	88.01 ± 0.42
		Time	0.91 ± 0.01	0.90 ± 0.01	0.94 ± 0.01	0.96 ± 0.01	0.80 ± 0.01
	5	Acc	89.83 ± 0.21	**89.87 ± 0.24**	89.81 ± 0.23	89.63 ± 0.13	89.14 ± 0.51
		Time	1.48 ± 0.01	1.47 ± 0.01	1.57 ± 0.01	1.52 ± 0.01	1.41 ± 0.01
Sector	1	Acc	**94.17 ± 0.27**	93.96 ± 0.36	94.08 ± 0.30	93.52 ± 0.41	93.57 ± 0.37
		Time	0.58 ± 0.01	0.56 ± 0.01	0.59 ± 0.01	0.56 ± 0.01	0.48 ± 0.01
	3	Acc	94.98 ± 0.24	**95.06 ± 0.28**	95.02 ± 0.27	94.78 ± 0.23	94.74 ± 0.17
		Time	1.69 ± 0.01	1.64 ± 0.01	1.71 ± 0.01	1.67 ± 0.01	1.40 ± 0.01
	5	Acc	**95.10 ± 0.27**	95.07 ± 0.26	95.06 ± 0.26	94.98 ± 0.22	94.95 ± 0.18
		Time	2.82 ± 0.02	2.69 ± 0.06	2.81 ± 0.01	2.78 ± 0.01	2.32 ± 0.02

all samples, which was 5690 for News20 and 1049 for Sector. Results are given in Fig. 2. While all methods scale roughly linearly with respect to k, we found that exact methods scaled very well, even on datasets with a large number of classes such as Sector.



(a) News20 (b) Sector

Fig. 2: Impact of the number of classes on training time.

B. *Solving the restricted problem exactly vs. approximately*

Previous works (e.g., [3], [5]) usually picked one way of solving the restricted problem and did not compare with others. Therefore, it is not clear yet whether it is more advantageous to solve the restricted problem exactly or approximately. To answer this question, we compared exact methods (Sort, Pivot) to bisection. As indicated in Algorithm 3, bisection requires a tolerance parameter τ for its stopping criterion. Choosing τ is a matter of trade-off: the smaller τ, the more accurately we solve the restricted problem but also the more iterations it takes. In our experiments, we ran bisection with different tolerance values: $\tau = 10^r$ where $r = -5$ (very strict), $r = -4$, $\ldots$, $r = -1$ (very loose).

In a first experiment, we compared exact methods and bisection in terms of convergence. Results are indicated in Fig. 3. With respect to the dual objective value, exact methods constantly outperformed bisection, even with $r = -5$ (very strict). With respect to test accuracy, our results indicate that setting the tolerance parameter too loose ($r \geq -2$) results in slower convergence and worse accuracy than exact methods. We found that the difference between the Sort and Pivot algorithms was usually small. This is due to the fact that the $O(k)$ complexity of Pivot is in expectation whereas the $O(k \log k)$ complexity of Sort is in worst case. We expect the Pivot method to be more beneficial when k is very large.

In a second experiment, we compared exact methods and bisection in terms of their robustness with respect to the regularization parameter C. Results are indicated in Fig. 4. Our results show that as C gets larger (less regularized), loosely solving the restricted problem results in poorer and poorer accuracy. Therefore, the larger C (the less regularized) the more important it becomes to solve the restricted problem accurately or exactly. However, bisection typically becomes slower than exact methods when $r \leq -2$. Therefore, we find that exact methods are the most cost-effective: they both find an exact solution and are computationally cheap.

VI. Conclusion

We presented a method for training multiclass SVMs by Euclidean projection onto the positive simplex. Our method allows to solve dual decomposition's restricted problem exactly in expected $O(k)$ time. We demonstrated on several large-scale high-dimensional datasets that solving the restricted problem exactly i) achieves state-of-the-art convergence ii) outperforms approximate methods for one-pass or few-pass training iii) scales well with respect to the number of classes.

Acknowledgment

This work was partially supported by the FIRST program.

REFERENCES

[1] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the 5th Conference on Learning Theory (COLT)*, 1992, pp. 144–152.

[2] J. Weston and C. Watkins, "Support vector machines for multi-class pattern recognition," in *Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 1999, pp. 219–224.

[3] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 265–292, 2002.

[4] Y. Lee, Y. Lin, and G. Wahba, "Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data," *Journal of the American Statistical Association*, vol. 99, pp. 67–81, 2004.

[5] S. S. Keerthi, S. Sundararajan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin, "A sequential dual method for large scale multi-class linear svms," ser. KDD '08, 2008, pp. 408–416.

[6] B. P., S. K. Shevade, S. Sundararajan, and S. S. Keerthi, "A sequential dual method for structural svms," in *Proceedings of the Eleventh SIAM International Conference on Data Mining*, 2011, pp. 223–234.

[7] M. Collins, A. Globerson, T. Koo, X. Carreras, and P. L. Bartlett, "Exponentiated gradient algorithms for conditional random fields and max-margin markov networks," *Journal of Machine Learning Research*, vol. 9, pp. 1775–1822, 2008.

[8] T. Joachims, T. Finley, and C.-N. J. Yu, "Cutting-plane training of structural svms," *Machine Learning*, vol. 77, no. 1, pp. 27–59, 2009.

[9] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: primal estimated sub-gradient solver for SVM," *Mathematical Programming*, pp. 1–28, 2010.

[10] A. Bordes, L. Bottou, P. Gallinari, and J. Weston, "Solving multiclass support vector machines with larank," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 89–96.

[11] J. C. Platt, "Advances in kernel methods." MIT Press, 1999, ch. Fast training of support vector machines using sequential minimal optimization, pp. 185–208.

[12] S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher, "Block-coordinate frank-wolfe optimization for structural svms," in *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[13] C.-J. Lin, "A formal analysis of stopping criteria of decomposition methods for support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1045–1052, 2002.

[14] K. Crammer and Y. Singer, "On the learnability and design of output codes for multiclass problems," *Machine Learning*, vol. 47, no. 2-3, pp. 201–233, 2002.

[15] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the l1-ball for learning in high dimensions," in *Proceedings of the International Conference on Machine Learning*, 2008, pp. 272–279.

[16] S. Shalev-Shwartz and Y. Singer, "Efficient learning of label ranking by soft projections onto polyhedra," *Journal of Machine Learning Research*, pp. 1567–1599, 2006.

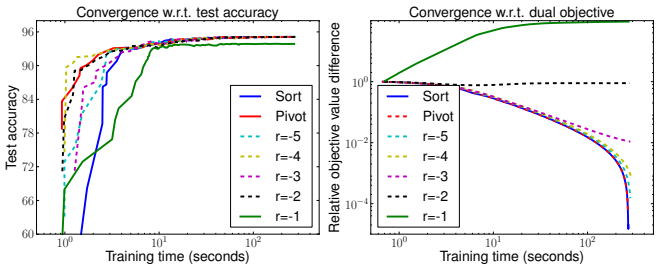
[17] J. Liu and J. Ye, "Efficient euclidean projections in linear time," in *Proceedings of the International Conference on Machine Learning*, 2009, pp. 657–664.

[18] M. Blondel, Y. Kubo, and U. Naonori, "Online passive-aggressive algorithms for non-negative matrix factorization and completion," in *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, 2014, pp. 96–104.

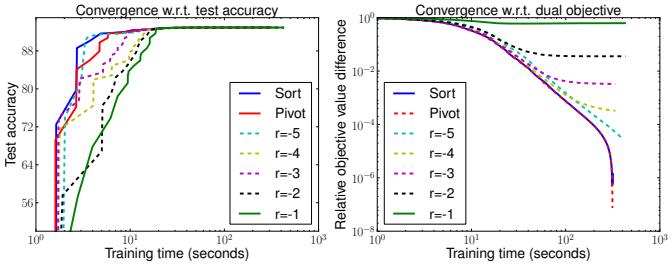
[19] M. Dredze, K. Crammer, and F. Pereira, "Confidence-weighted linear classification," in *Proceedings of International Conference on Machine Learning (ICML)*, 2008, pp. 264–271.

[20] M. Blondel, K. Seki, and K. Uehara, "Block coordinate descent algorithms for large-scale sparse multiclass classification," *Machine Learning*, vol. 93, no. 1, pp. 31–52, 2013.

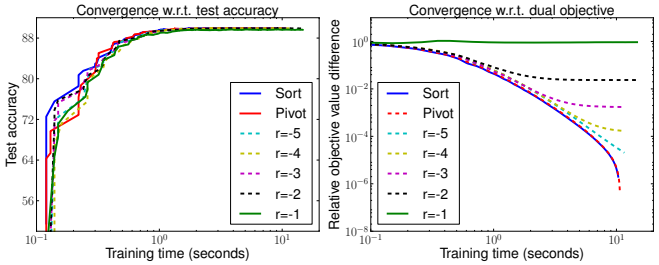
[21] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proceedings of International Conference on Machine Learning (ICML)*, 2009, pp. 1113–1120.



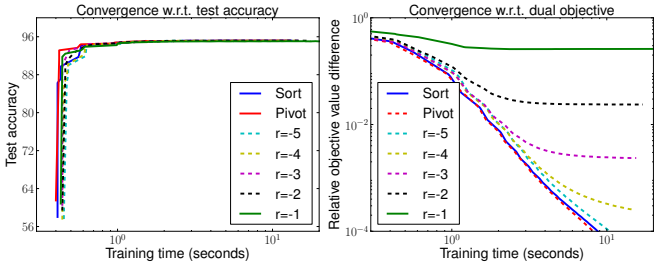
(a) Amazon7



(b) RCV1



(c) News20



(d) Sector

Fig. 3: Convergence of exact methods (Sort, Pivot) and bisection with tolerance parameter $\tau = 10^r$, where $r = -5$ (very strict), $r = -4$, ..., $r = -1$ (very loose).

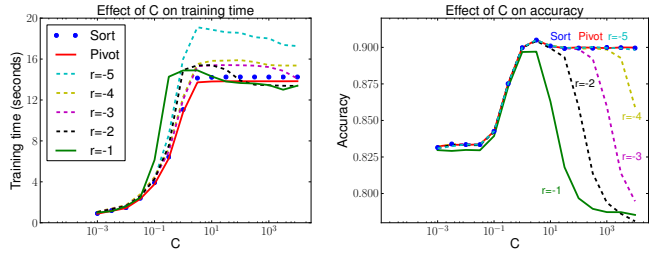


Fig. 4: Comparison between exact methods (Sort, Pivot) and bisection when varying the regularization parameter C on the News20 dataset. For bisection, we set the tolerance parameter of the stopping criterion to $\tau = 10^r$.

Supplementary material

We briefly present how to implement the SMO and Frank-Wolfe methods below.

A. Sequential Minimal Optimization (SMO)

Sequential Minimal Optimization (SMO) was used to solve the restricted problem in Eq. (6) in LaRank [10]. Suppose that we choose two coordinates $p \in [k]$ and $q \in [k] \setminus \{p\}$. The main idea of SMO is that we can maintain feasibility with respect to the equality constraint $\sum_{m=1}^{k} \alpha_i^m = 0$ if we update α_i^p and α_i^q by $\alpha_i^p \leftarrow \alpha_i^p + \lambda$ and $\alpha_i^q \leftarrow \alpha_i^q - \lambda$, respectively. Note that this is equivalent to setting $\boldsymbol{\delta}_i$ in Eq. (6) to $[0, 0, \ldots, \underbrace{\lambda}_{p}, \ldots, \underbrace{-\lambda}_{q}, \ldots, 0, 0]^{\mathrm{T}}$. In that case, Eq. (6) simplifies to

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & \|\boldsymbol{x}_i\|^2 \lambda^2 + (g_i^p - g_i^q)\lambda \\ \text{subject to} \quad & \alpha_i^q - C_i^q \leq \lambda \leq C_i^p - \alpha_i^p. \end{aligned}$$

This is a univariate quadratic objective with a box-constraint. Therefore, the optimal solution is obtained by solving the unconstrained problem followed by a projection to the box:

$$\lambda = \min\Big(\max\Big(\frac{g_i^q - g_i^p}{2\|\boldsymbol{x}_i\|^2}, \alpha_i^q - C_i^q\Big), C_i^p - \alpha_i^p\Big).$$

We then maintain $\boldsymbol{w}_p$ and $\boldsymbol{w}_q$ by $\boldsymbol{w}_p \leftarrow \boldsymbol{w}_p + \lambda \boldsymbol{x}_i$ and $\boldsymbol{w}_q \leftarrow \boldsymbol{w}_q - \lambda \boldsymbol{x}_i$, respectively. Motivated by Eq. (5) and following [5], we choose $p = \underset{m \in [k]}{\operatorname{argmax}}\, g_i^m$ and $q \in \{m : \alpha_i^m < C_i^m\}$ cyclically.

B. Frank-Wolfe method

Recently, [12] proposed a block Frank-Wolfe algorithm for structured SVMs. The main idea of the Frank-Wolfe method is to solve a constrained linear approximation of the objective function. In our notation, this leads to the following sub-problem:

$$\begin{aligned} \underset{\boldsymbol{s}}{\text{minimize}} \quad & \boldsymbol{g}_i^{\mathrm{T}} \boldsymbol{s} \\ \text{subject to} \quad & \boldsymbol{s} \leq \boldsymbol{C}_i \\ & \boldsymbol{s}^{\mathrm{T}} \mathbf{1} = 0. \end{aligned}$$

Then, we can update $\boldsymbol{\alpha}_i$ by $\boldsymbol{\alpha}_i \leftarrow \boldsymbol{\alpha}_i + \gamma(\boldsymbol{s} - \boldsymbol{\alpha}_i)$ or equivalently $\boldsymbol{\alpha}_i \leftarrow (1-\gamma)\boldsymbol{\alpha}_i + \gamma \boldsymbol{s}$, where $\gamma \in [0, 1]$. The optimal solution to this problem (different from [12] because they use a different dual formulation) is $\boldsymbol{s} = \boldsymbol{C}_i - C\boldsymbol{e}_j = C(\boldsymbol{e}_{y_i} - \boldsymbol{e}_j)$ where

$$j = \underset{m \in [k]}{\operatorname{argmax}}\, g_i^m = \underset{m \in [k]}{\operatorname{argmax}}\, \boldsymbol{w}_m^{\mathrm{T}} \boldsymbol{x}_i + \Delta_i^m$$

In other words, $\boldsymbol{s} = \mathbf{0}$ if $j = y_i$, otherwise $\boldsymbol{s}$ is zero everywhere except in the y_i^{th} position where it is C and in the j^{th} position where it is $-C$. Recall that for an update of the form $\boldsymbol{\alpha}_i \leftarrow \boldsymbol{\alpha}_i + \boldsymbol{\delta}_i$, we need to solve Eq. (6). To find γ, we can therefore inject $\boldsymbol{\delta}_i = \gamma(\boldsymbol{s} - \boldsymbol{\alpha}_i)$ in the objective of Eq. (6), which leads to the following optimization problem:

$$\underset{\gamma \in [0,1]}{\text{minimize}} \quad \frac{\|\boldsymbol{x}_i\|^2}{2} \gamma^2 ||\boldsymbol{s} - \boldsymbol{\alpha}_i||^2 + \gamma\, \boldsymbol{g}_i^{\mathrm{T}}(\boldsymbol{s} - \boldsymbol{\alpha}_i). \tag{8}$$

The constraint $\gamma \in [0, 1]$ ensures that the constraints of the dual objective will be satisfied. The problem in Eq. (8) is a univariate one and has a simple closed-form solution:

$$\gamma = \min\Big(\max\Big(\frac{\boldsymbol{g}_i^{\mathrm{T}}(\boldsymbol{\alpha}_i - \boldsymbol{s})}{\|\boldsymbol{x}_i\|^2 \|\boldsymbol{s} - \boldsymbol{\alpha}_i\|^2}, 0\Big), 1\Big).$$