Toon Calders Floriana Esposito
Eyke Hüllermeier Rosa Meo (Eds.)

LNAI 8724

Machine Learning and Knowledge Discovery in Databases

European Conference, ECML PKDD 2014
Nancy, France, September 15–19, 2014
Proceedings, Part I

1 Part I

Springer

FASOLE: Fast Algorithm for Structured Output LEarning

Vojtech Franc

Czech Technical University in Prague
Faculty of Electrical Engineering
Technicka 2, 166 27 Prague 6, Czech Republic
xfrancv@cmp.felk.cvut.cz

Abstract. This paper proposes a novel Fast Algorithm for Structured Ouput LEarning (FASOLE). FASOLE implements the sequential dual ascent (SDA) algorithm for solving the dual problem of the Structured Output Support Vector Machines (SO-SVM). Unlike existing instances of SDA algorithm applied for SO-SVM, the proposed FASOLE uses a different working set selection strategy which provides nearly maximal improvement of the objective function in each update. FASOLE processes examples in an on-line fashion and it provides certificate of optimality. FASOLE is guaranteed to find the ε-optimal solution in $\mathcal{O}(\frac{1}{\varepsilon^2})$ time in the worst case. In the empirical comparison FASOLE consistently outperforms the existing state-of-the-art solvers, like the Cutting Plane Algorithm or the Block-Coordinate Frank-Wolfe algorithm, achieving up to an order of magnitude speedups while obtaining the same precise solution.

1 Introduction

The Structured Output Support Vector Machines SO-SVM [17,19] is a supervised algorithm for learning parameters of a linear classifiers with possibly exponentially large number of classes. SO-SVM translate learning into a convex optimization problem size of which scales with the number of classes which rules out application of common off-the-shelf solvers. The specialized solvers can be roughly split to batch methods and on-line solvers. The batch methods, like variants of the Cutting Plane Algorithm (CPA) [18,8] or the column generation algorithm [19], approximate the SO-SVM objective by an iteratively built global under-estimator called the cutting plane model (the column generation algorithm instead approximates the feasible set). Optimizing the cutting plane model is cheaper than the original problem and, in addition, it provides a certificate of optimality. The bottle-neck of the CPA is the expensive per-iteration computational complexity. Namely, computation of a single element (the cutting plane) of the cutting plane model requires calling the *classification oracle* on all training examples. Note that in structured setting the classification is often time consuming. Moreover, many iterations are typically needed before the cutting plane model becomes tight and the approximate solution sufficiently precise.

The on-line methods, like the Stochastic Gradient Descent (SGD) [13,15], process the training examples one by one with a cheap update requiring a single call of the classification oracle. A disadvantage of the SGD is its sensitivity to setting of the step size

T. Calders et al. (Eds.): ECML PKDD 2014, Part I, LNCS 8724, pp. 402–417, 2014.

and a missing clear stopping condition as the method does not provide a certificate of optimality. Unlike the SGD which optimizes the primal objective, the recently proposed Sequential Dual Method for Structural SVMs (SDM) [1] and the Block-Coordinate Frank-Wolfe (BCFW) [9] are on-line solvers maximizing the Lagrange dual of the SO-SVM problem. SDM and BCFW are instances of the same optimization framework in the sequel denoted as the Sequential Dual Ascent (SDA) method. The SDA methods iteratively update blocks of dual variables each of them being associated with a single training example. Particular instances of SDA method, like SDM or BCFW, differ in the strategy used to select the working set containing the dual variables to be updated. The SDA methods if compared to the SGD algorithm have two main advantages. First, for fixed working set the optimal step size can be computed analytically. Second, the optimized dual objective provides a certificate of optimality useful for defining a rigorous stopping condition.

The convergence speed of the SDA methods is largely dependent on the working set selection strategy. In this paper, we propose a novel SDA algorithm for solving the SO-SVM dual using a working set selection strategy which yields nearly maximal improvement of the dual objective in each iteration. We named the proposed solver as the Fast Algorithm for Structured Ouput LEarning (FASOLE). The same idea has been previously applied for optimization of simpler QP tasks emerging in learning of two-class SVM classifiers with L2-hinge loss [4,5] and L1-hinge loss [3,6]. The SDA solver using a similar working set selection strategy is implemented for example in popular LibSVM [2]. Our paper extends these solvers to the structured output setting. The structured output setting imposes several difficulties, namely, the SO-SVM dual problem has exponentially large number of variables and m linear equality constraints in contrast to the two-class SVM dual having only m variables and single equality constraint. The extreme size of the SO-SVM does not permit operations feasible in two-class case like maintaining all dual variables and buffering the columns of the Hessian matrix. The proposed method thus introduces a sparse representation of the SO-SVM dual and a set of heuristics to reflect the mentioned difficulties. In addition, we provide a novel convergence analysis which guarantees that the proposed SDA solver finds ε-optimal solution in $\mathcal{O}(\frac{1}{\varepsilon^2})$ time. We experimentally compare the proposed FASOLE against BCFW, SDM and CPA showing that FASOLE consistently outperforms all competing methods achieving up to an order of magnitude speedup. We remark that recently proposed BCFW and SDM have not been compared so far hence their empirical study is an additional contribution of this paper.

The paper is organized as follows. The problem to be solved is formulated in Section 2. Section 3 describes the proposed solver. Relation to existing methods is discussed in Section 4. Section 5 presents experiments and Section 6 concludes the paper.

2 Formulation of the Problem

Let us consider a linear classifier $h\colon \mathcal{X} \times \mathbb{R}^n \to \mathcal{Y}$ defined as

$$h(x; \boldsymbol{w}) \in \operatorname*{Argmax}_{y \in \mathcal{Y}} \langle \boldsymbol{w}, \boldsymbol{\psi}(x, y) \rangle$$

which assigns label $y \in \mathcal{Y}$ to observations $x \in \mathcal{X}$ according to a linear scoring function given by a scalar product between a feature vector $\boldsymbol{\psi}\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^n$ and a parameter vector $\boldsymbol{w} \in \mathbb{R}^n$ to be learned from data. In the structured output setting the label set $\mathcal{Y}$ is finite but typically very large, e.g. $\mathcal{Y}$ contains all possible image segmentations. Given a set of examples $\{(x_1, y_1), \ldots, (x_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$, the SO-SVM algorithm translates learning of the parameters $\boldsymbol{w} \in \mathbb{R}^n$ into the following convex problem

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}^n} P(\boldsymbol{w}) := \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum_{i \in \mathcal{I}} \max_{y \in \mathcal{Y}} \big(\Delta_i(y) + \langle \boldsymbol{w}, \boldsymbol{\psi}_i(y)\rangle\big) \tag{1}$$

where $\mathcal{I} = \{1, \ldots, m\}$, $\boldsymbol{\psi}_i(y) = \boldsymbol{\psi}(x_i, y) - \boldsymbol{\psi}(x_i, y_i)$, $\Delta_i(y) = \Delta(y_i, y)$ is a loss function and $\lambda > 0$ is a regularization constant. For a convenience of notation we assume that the loss function satisfies $\Delta(y, y) = 0$, $\forall y \in \mathcal{Y}$, which implies that $\Delta_i(y_i) + \langle \boldsymbol{w}, \boldsymbol{\psi}_i(y_i)\rangle = 0$. Note that all common loss functions like e.g. Hamming loss satisfy this assumption. The problem (1) can be equivalently expressed as a quadratic program whose Lagrange dual, denoted as SO-SVM dual, reads

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha} \in \mathcal{A}} D(\boldsymbol{\alpha}) := \langle \boldsymbol{b}, \boldsymbol{\alpha}\rangle - \frac{1}{2}\|\boldsymbol{A}\boldsymbol{\alpha}\|^2 , \tag{2}$$

where $\boldsymbol{\alpha} = (\alpha_i(y) \mid i \in \mathcal{I}, y \in \mathcal{Y}) \in \mathbb{R}^d$ is vector of $d = m|\mathcal{Y}|$ dual variables, $\boldsymbol{b} = (\Delta_i(y) \mid i \in \mathcal{I}, y \in \mathcal{Y}) \in \mathbb{R}^d$ is a vector containing losses on training examples, $\boldsymbol{A} = (\boldsymbol{\psi}_i(y)/\sqrt{\lambda} \mid i \in \mathcal{I}, y \in \mathcal{Y}) \in \mathbb{R}^{n \times d}$ is a matrix of feature vectors, and $\mathcal{A} = \big\{\boldsymbol{\alpha} \in \mathbb{R}^d \mid \boldsymbol{\alpha} \geq 0 \;\wedge\; \sum_{y \in \mathcal{Y}} \alpha_i(y) = \frac{1}{m}\ , i \in \mathcal{I}\big\}$ denotes a feasible set. Since the primal problem (1) is convex and non-degenerate, the duality gap at the optimum is zero, i.e. $P(\boldsymbol{w}^*) = D(\boldsymbol{\alpha}^*)$. The optimal primal variables can be computed from the optimal dual variables by $\boldsymbol{w}^* = -\frac{1}{\sqrt{\lambda}}\boldsymbol{A}\boldsymbol{\alpha}^*$.

In this paper we propose a solver which iteratively maximizes the SO-SVM dual (2). Although maintaining the dual problem in computer memory is not feasible, an approximate solution can be found thanks to the problem sparsity. For any prescribed $\varepsilon > 0$, our solver finds an ε-optimal solution $\hat{\boldsymbol{w}}$ satisfying $P(\hat{\boldsymbol{w}}) \leq P(\boldsymbol{w}^*) + \varepsilon$ in time not bigger than $\mathcal{O}(\frac{1}{\varepsilon^2})$. Our solver is modular: it accesses the problem only via a classification oracle solving so called loss-augmented inference task, i.e. for given $i, \boldsymbol{w}$ the classification oracle returns an optimal solution and the optimal value of

$$\max_{y \in \mathcal{Y}} \big(\Delta_i(y) + \langle \boldsymbol{w}, \boldsymbol{\psi}_i(y)\rangle\big) . \tag{3}$$

3 Proposed Algorithm Solving SO-SVM Dual

In this section we describe the proposed solver. For the sake of space we put derivations and proofs to a supplementary material available online[1].

[1] ftp://cmp.felk.cvut.cz/pub/cmp/articles/franc/Franc-FasoleSupplementary-ECML2014.pdf

3.1 Generic SDA

In this section we first outline the idea of a generic SDA algorithm for solving SO-SVM dual (2). In the next section we then describe the proposed instance of the generic SDA.

A generic SDA algorithm converts optimization of the SO-SVM dual (2) into a series of simpler auxiliary QP tasks solvable analytically. Starting from a feasible point $\boldsymbol{\alpha} \in \mathcal{A}$, an SDA algorithm iteratively applies the update rule

$$\boldsymbol{\alpha}^{\text{new}} := \operatorname*{argmax}_{\boldsymbol{\alpha} \in \mathcal{A}_L} D(\boldsymbol{\alpha}) \,, \tag{4}$$

where $\mathcal{A}_L \subset \mathcal{A}$ is a line between the current solution $\boldsymbol{\alpha}$ and a point $\boldsymbol{\beta}$ selected from $\mathcal{A}$,

$$\mathcal{A}_L = \{\boldsymbol{\alpha}' \mid \boldsymbol{\alpha}' = (1-\tau)\boldsymbol{\alpha} + \tau\boldsymbol{\beta} \,, \tau \in [0,1]\} \,. \tag{5}$$

The update rule (4) is an auxiliary QP task having the same objective as the original SO-SVM dual but the feasible set is reduced to a line inside $\mathcal{A}$. This implies that the new solution $\boldsymbol{\alpha}^{\text{new}}$ is also feasible. Let us define a single-variable quadratic function

$$D_L(\tau) = D\big((1-\tau)\boldsymbol{\alpha} + \tau\boldsymbol{\beta}\big)$$

corresponding to the dual objective $D(\boldsymbol{\alpha})$ restricted to the line $\mathcal{A}_L$. If the point $\boldsymbol{\beta}$ is selected such that the derivative of $D(\boldsymbol{\alpha})$ along the line $\mathcal{A}_L$ evaluated at $\boldsymbol{\alpha}$ is positive, i.e. $D_L(0)' > 0$, then the update (4) strictly increases the objective function, i.e. $D(\boldsymbol{\alpha}^{\text{new}}) - D(\boldsymbol{\alpha}) > 0$ holds. Moreover, the update (4) has a simple analytical solution (supplementary material, Sec 1.1)

$$\boldsymbol{\alpha}^{\text{new}} := (1-\tau)\boldsymbol{\alpha} + \tau\boldsymbol{\beta}$$

where

$$\tau := \operatorname*{argmax}_{\tau' \in [0,1]} D_L(\tau') = \min\left(1, \frac{\langle \boldsymbol{\beta} - \boldsymbol{\alpha}, \boldsymbol{b} - \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{\alpha}\rangle}{\langle \boldsymbol{\alpha} - \boldsymbol{\beta}, \boldsymbol{A}^T\boldsymbol{A}(\boldsymbol{\alpha} - \boldsymbol{\beta})\rangle}\right) \,. \tag{6}$$

Algorithm 1. Generic SDA for solving SO-SVM dual (2)

```
Initialization: select a feasible $\boldsymbol{\alpha} \in \mathcal{A}$
repeat
    Select $\boldsymbol{\beta} \in \mathcal{A}$ such that $D_L(0)' > 0$.
    Compute $\tau$ by (6).
    Compute the new solution $\boldsymbol{\alpha} := (1-\tau)\boldsymbol{\alpha} + \tau\boldsymbol{\beta}$.
until until convergence;
```

Algorithm (1) outlines the generic SDA algorithm. The recently proposed SDM [1] and the BCFW Algorithm [9] are instances of the generic SDA which differ in the way how they construct the point $\boldsymbol{\beta}$. Note, that the point $\boldsymbol{\beta}$ determins which variables will be modified (the working set) by the update rule. For example, the BCFW simultaneously updates all $|\mathcal{Y}|$ dual variables associated with one training example, while, the method

SDM updates just two variables at a time. In Section 4 we describe the related instances of the generic SDA in more details.

Our algorithm is also instance of the generic SDA which adopts a different selection strategy originally proposed for two-class SVM solvers independently proposed by [3] and [4]. This method, in [3] coined the Working Set Selection strategy using second order information (WSS2), selects two variables ensuring nearly maximal improvement of the objective. In the next section we describe the adaptation of the WSS2 to solving the SO-SVM dual.

3.2 Proposed SDA Solver with WSS2 Selection Strategy

The proposed solver constructs the point $\boldsymbol{\beta}$ as follows

$$\beta_j(y) := \begin{cases} \alpha_i(u) + \alpha_i(v) & \text{if} \quad j = i \wedge y = u \\ 0 & \text{if} \quad j = i \wedge y = v \\ \alpha_j(y) & \text{otherwise} \end{cases} \tag{7}$$

where $(i, u, v) \in \mathcal{I} \times \mathcal{Y} \times \mathcal{Y}$ is a triplet such that $u \neq v$ (the way how (i, u, v) is selected will be described below). We denote the SDA update rule (4) using $\boldsymbol{\beta}$ constructed by (7) as the Sequential Minimal Optimization (SMO) rule [7]. The SMO rule changes the minimal number of dual variables, in our case two, without escaping the feasible set $\mathcal{A}$. There are $m|\mathcal{Y}|(|\mathcal{Y}| - 1)$ SMO rules in total out of which we need to select one in each iteration. A particular SMO rule determined by the choice of (i, u, v) changes the variables $\alpha_i(u)$ and $\alpha_i(v)$ associated with labels u and v and the i-th example. We now describe our strategy to select (i, u, v).

Selection of i. Recall that the ultimate goal is to find an ε-optimal solution of the SO-SVM dual (2). Specifically, we aim to find a primal-dual pair $(\boldsymbol{w}, \boldsymbol{\alpha}) \in \mathbb{R}^n \times \mathcal{A}$ such that the duality gap is at most ε, i.e. $G(\boldsymbol{w}, \boldsymbol{\alpha}) = P(\boldsymbol{w}) - D(\boldsymbol{\alpha}) \leq \varepsilon$ holds. Let us define shorthands

$$\boldsymbol{w} = -\frac{1}{\sqrt{\lambda}} \boldsymbol{A}\boldsymbol{\alpha} \qquad \text{and} \qquad s_i(y, \boldsymbol{\alpha}) = \Delta_i(y) + \langle \boldsymbol{w}, \boldsymbol{\psi}_i(y) \rangle$$

for the primal solution $\boldsymbol{w}$ constructed from the dual solution $\boldsymbol{\alpha}$ and the score function $s_i(y, \boldsymbol{w})$ of the classification oracle (3) with parameters $\boldsymbol{w}$, respectively. Using this notation it is not difficult to show (supplementary material, Sec. 1.3) that the duality gap can be computed by the following formula

$$G(\boldsymbol{w}, \boldsymbol{\alpha}) = \frac{1}{m} \sum_{i \in \mathcal{I}} G_i(\boldsymbol{w}, \boldsymbol{\alpha})$$

where

$$G_i(\boldsymbol{w}, \boldsymbol{\alpha}) = \max_{y \in \mathcal{Y}} s_i(y, \boldsymbol{w}) - m \sum_{y \in \mathcal{Y}} \alpha_i(y) s_i(y, \boldsymbol{w}) \, .$$

The proposed solver goes through the examples and it uses the value of $G_i(\boldsymbol{w}, \boldsymbol{\alpha})$ to decide whether the block of variables $\boldsymbol{\alpha}_i = (\alpha_i(y) \mid y \in \mathcal{Y})$, associated with the i-th example, should be updated. In particular, if $G_i(\boldsymbol{w},\boldsymbol{\alpha}) > \varepsilon$ holds then $\boldsymbol{\alpha}_i$ is updated otherwise they are skipped and the next block of variables is checked. It is clear that if $G_i(\boldsymbol{w},\boldsymbol{\alpha}) \leq \varepsilon$ holds for all $i \in \mathcal{I}$, i.e. no variables are selected for update, the target duality gap $G(\boldsymbol{w},\boldsymbol{\alpha}) \leq \varepsilon$ has been already achieved and thus no update is needed.

Selection of u and v. Here we employ the WSS2 strategy of [3,4]. Given block of variables $\boldsymbol{\alpha}_i$, our goal is to select among them two, $\alpha_i(u)$ and $\alpha_i(v)$, which if used in the SMO rule will cause a large increment, $\delta(i,u,v) = D(\boldsymbol{\alpha}^{\text{new}}) - D(\boldsymbol{\alpha})$, of the dual objective. First, we need to identify those pairs (u,v) which guarantee positive change of dual objective, i.e. those for which $D'_L(0) > 0$ holds. Using (7) the condition $D'_L(0) > 0$ can be written as (supplementary material, Sec. 1.2)

$$\alpha_i(v)\big(s_i(u,\boldsymbol{w}) - s_i(v,\boldsymbol{w})\big) > 0 \,. \tag{8}$$

Provided the condition (8) holds, application of the SMO rule given by a triplet (i,u,v) increases the dual objective exactly by the quantity (supplementary material, Sec. 1.2)

$$\delta(i,u,v) = \begin{cases} \dfrac{\lambda(s_i(u,\boldsymbol{w}) - s_i(v,\boldsymbol{w}))}{\|\boldsymbol{\psi}_i(u) - \boldsymbol{\psi}_i(v)\|^2} & \text{if } \tau < 1 \\ \alpha_i(v)(s_i(u,\boldsymbol{w}) - s_i(v,\boldsymbol{w})) - \dfrac{\alpha_i(v)^2}{2\lambda}\|\boldsymbol{\psi}_i(u) - \boldsymbol{\psi}_i(v)\|^2 & \text{if } \tau = 1 \end{cases} \tag{9}$$

The optimal strategy would be to find the pair (u,v) maximizing $\delta(i,u,v)$, however, this is not feasible due to a large number of candidate pairs $(u,v) \in \mathcal{Y} \times \mathcal{Y}$. Instead we use a cheaper WSS2 strategy which has been shown to yield nearly the same improvement as trying all pairs [4]. We first find $\hat{u}$ by maximizing the dominant term $s_i(u,\boldsymbol{w})$ appearing in the improvement formulas. This corresponds to fining the most violated primal constraint associated with i-th example by solving

$$\hat{u} \in \operatorname*{argmax}_{y\in\mathcal{Y}} s_i(y,\boldsymbol{w}) \tag{10}$$

via using the classification oracle. When $\hat{u}$ is fixed, we find $\hat{v}$ which brings the maximal improvement by

$$\hat{v} \in \operatorname*{argmax}_{y\in\mathcal{Y}_i} \delta(i,\hat{u},y) \tag{11}$$

where $\mathcal{Y}_i = \{y \in \mathcal{Y} \mid \alpha_i(y) > 0\}$ is a set of labels corresponding to non-zero dual variables associated with the i-th example. Note that the maximization tasks (10) and (11) do not need to have a unique solution in which case we take any maximizer.

The SDA solver using WSS2 is summarized in Algorithm 2. Note that the SDA-WSS2 algorithm is an on-line method passing thought the examples and updating those blocks of variables found to be sub-optimal. The algorithm stops if $G_i(\boldsymbol{w},\boldsymbol{\alpha}) \leq \varepsilon$, $\forall i \in \mathcal{I}$, implying that the duality gap $G(\boldsymbol{w},\boldsymbol{\alpha})$ is not larger than ε. Besides primal variables $\boldsymbol{w} \in \mathbb{R}^n$ the algorithm maintains the non-zero dual variables, $\alpha_i(y) > 0$, $y \in \mathcal{Y}_i$, their number is upper bounded by the number of updates. Although the primal variables are at any time related to the dual variables by $\boldsymbol{w} = -\frac{1}{\sqrt{\lambda}}\boldsymbol{A\alpha}$ it is beneficial to maintain the vector $\boldsymbol{w}$ explicitly as it speeds up the computation of the score

$$s_i(y,\boldsymbol{w}) = \Delta_i(y) + \langle \boldsymbol{\psi}_i(y), \boldsymbol{w}\rangle = \Delta_i(y) - \frac{1}{\lambda}\sum_{j\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}_i} \alpha_j(y')\langle \boldsymbol{\psi}_i(y), \boldsymbol{\psi}_j(y')\rangle \,.$$

Algorithm 2. SDA-WSS2 algorithm for solving SO-SVM dual (2)

```
Input: precision parameter $\varepsilon > 0$, regularization constant $\lambda > 0$
Output: $\varepsilon$-optimal primal-dual pair $(\boldsymbol{w}, \boldsymbol{\alpha})$
Initialization:
$\boldsymbol{w} = \boldsymbol{0}$, $\mathcal{Y}_i = \{y_i\}$, $\alpha_i(y) = \begin{cases} \frac{1}{m} \text{ if } & y = y_i \\ 0 \text{ otherwise} \end{cases}, i \in \mathcal{I}$
repeat
    num_updates := 0
    forall the $i \in \mathcal{I}$ do
        $u_1 := \operatorname{argmax}_{y \in \mathcal{Y}} s_i(y, \boldsymbol{w})$
        $u_2 := \operatorname{argmax}_{y \in \mathcal{Y}_i} s_i(y, \boldsymbol{w})$
        if $s_i(u_1, \boldsymbol{w}) > s_i(u_2, \boldsymbol{w})$ then
            $\hat{u} := u_1$
        else
            $\hat{u} := u_2$
        if $s_i(\hat{u}, \boldsymbol{w}) - m \sum_{y \in \mathcal{Y}_i} \alpha_i(y) s_i(y, \boldsymbol{w}) > \varepsilon$ then
            num_updates := num_updates + 1
            $\hat{v} := \underset{y \in \{y' \in \mathcal{Y}_i \mid \alpha_i(y') > 0\}}{\operatorname{argmax}} \delta(i, \hat{u}, y)$
            $\tau := \min \left\{ 1, \frac{\lambda(s_i(\hat{u}, \boldsymbol{w}) - s_i(\hat{v}, \boldsymbol{w}))}{\alpha_i(\hat{v}) \|\boldsymbol{\psi}_i(\hat{u}) - \boldsymbol{\psi}_i(\hat{v})\|^2} \right\}$
            $\boldsymbol{w} := \boldsymbol{w} + (\boldsymbol{\psi}_i(\hat{v}) - \boldsymbol{\psi}_i(\hat{u})) \frac{\tau \alpha_i(\hat{v})}{\lambda}$
            $\alpha_i(\hat{u}) := \alpha_i(\hat{u}) + \tau \alpha_i(\hat{v})$
            $\alpha_i(\hat{v}) := \alpha_i(\hat{v}) - \tau \alpha_i(\hat{v})$
            if $\hat{u} = u_1$ then
                $\mathcal{Y}_i := \mathcal{Y}_i \cup \{u_1\}$
until num_updates = 0;
```

Note, however, that all computations can be carried out in terms of the dual variables $\boldsymbol{\alpha}$. This property allows to kernelize the algorithm by replacing $\langle \boldsymbol{\psi}_i(y), \boldsymbol{\psi}_j(y') \rangle$ with a selected kernel function.

The computational bottle neck of the SDA-WSS2 is calling the classification oracle to solve $u_1 := \operatorname{argmax}_{y \in \mathcal{Y}} s_i(y, \boldsymbol{w})$. The other maximization problems over $\mathcal{Y}_i$, which are required to select u_2 and v_2, can be solved exhaustively since the set $\mathcal{Y}_i$ contains only those $\boldsymbol{\psi}_i(y)$ corresponding to at least once updated dual variable $\alpha_i(y)$.

The convergence of the SDA-WSS2 is ensured by the following theorem:

Theorem 1. *For any $\varepsilon > 0$ and $\lambda > 0$, Algorithm 2 terminates after*

$$T = \frac{8LD^2}{\varepsilon^2 \lambda}$$

updates at most where $L = \max_{y \in \mathcal{Y}, y' \in \mathcal{Y}} \Delta(y, y')$ and $D = \max_{i \in \mathcal{I}, y \in \mathcal{Y}} \|\boldsymbol{\psi}(x_i, y)\|$.

Proof is given in the supplementary material, Section 2.

We point out without giving a formal proof that the convergence Theorem 1 is valid not only for the SDA-WSS2 algorithm but also for a broader class of SDA solvers using the SMO update rule. In particular, the idea behind the proof of Theorem 1 applies to the DCM of [1] for which no such bound has been published so far.

The competing methods like the BCFW [9] and the CPA [18] are known to converge to the ε-optimal solution in $\mathcal{O}(\frac{1}{\varepsilon})$ time which is order of magnitude better compared to our bound $\mathcal{O}(\frac{1}{\varepsilon^2})$ for the SDA-WSS2 algorithm. However, all the bounds are obtained by the worst case analysis and little is known about their tightness. The empirical simulations provided in Section 5 show that the actual number of updates required by the proposed algorithm is consistently much smaller (up to order of magnitude) compared to the competing methods.

3.3 FASOLE: Fast Algorithm for Structured Output LEarning

In this section we describe a set of heuristics significantly decreasing the absolute computational time of SDA-WSS2 without affecting its convergence guarantees. We denote SDA-WSS2 algorithm with the implemented heuristics as the Fast Algorithm for Structured Output LEarning (FASOLE). A pseudo-code of the FASOLE is summarized by Algorithm 3. The implemented heuristics aim at i) further reducing the number of oracle calls and ii) using a tighter estimate of the duality gap which is used as a certificate of the ε-optimality. In particular, FASOLE uses the following set of heuristics:

Reduced problem. FASOLE maintains vector $\hat{\boldsymbol{b}}$ and matrix $\hat{\boldsymbol{A}}$ containing a subset of coefficients $(\boldsymbol{b}, \boldsymbol{A})$ of the SO-SVM dual (2). The coefficients $(\hat{\boldsymbol{b}}, \hat{\boldsymbol{A}})$ correspond to the dual variables $\hat{\boldsymbol{\alpha}} = (\alpha_i(y) \mid y \in \mathcal{Y}_i, i \in \mathcal{I})$ which has been updated in the course of algorithm. The remaining variables are zero hence the corresponding coefficients need not be maintained. At the end of each pass through the examples we use $\hat{\boldsymbol{b}}$ and $\hat{\boldsymbol{A}}$ to find the optimal setting of $\hat{\boldsymbol{\alpha}}$ by solving a reduced dual problem

$$\hat{\boldsymbol{\alpha}} := \operatorname*{argmax}_{\boldsymbol{\alpha}' \in \mathcal{A}} \langle \hat{\boldsymbol{b}}, \boldsymbol{\alpha}' \rangle - \frac{1}{2} \|\hat{\boldsymbol{A}} \boldsymbol{\alpha}'\|^2 \tag{12}$$

We use the SDA-WSS2 Algorithm 2, "worm"-started from the current solution $\hat{\boldsymbol{\alpha}}$, to find ε-optimal solution of (12), i.e. FASOLE uses one loop optimizing over all variables and second loop for optimizing those variables which have been selected by the first loop. This strategy reduces the number oracle calls and is cheap due to the warm start.

Variable shrinking. SDA-WSS2 Algorithm 2 checks optimality of dual variables $\boldsymbol{\alpha}_i$ in each iteration irrespectively if they have been found optimal in the previous pass. FASOLE instead introduces binary flag, satisfied(i), which is set true if the corresponding variables $\boldsymbol{\alpha}_i$ already satisfied the partial duality gap constraint $G_i(\boldsymbol{w}, \boldsymbol{\alpha}) \leq \varepsilon$. Once all variables have been found optimal, i.e. $\text{satisfied(i)} = \text{true}$, $\forall i \in \mathcal{I}$, the flags are reset to false and the process starts again. This strategy allows to concentrate on non-optimal variables without wasting oracle calls on already optimal ones.

On-line and batch regime. The SDA-WSS2 Algorithm 2 stops if $G_i(\boldsymbol{w}, \boldsymbol{\alpha}) \leq \varepsilon$, $\forall i \in \mathcal{I}$, holds which implies $G(\boldsymbol{w}, \boldsymbol{\alpha}) \leq \varepsilon$. This stopping conditions is cheap and can be evaluated in an on-line manner, however, it is overly stringent because the actual duality gap $G(\boldsymbol{w}, \boldsymbol{\alpha})$ is usually a way below ε. In fact $G(\boldsymbol{w}, \boldsymbol{\alpha}) = \varepsilon$ holds only

Algorithm 3. FASOLE: Fast Algorithm for Structured Output LEarning

```
Input: precision parameter $\varepsilon > 0$, regularization constant $\lambda > 0$
Output: $\varepsilon$-optimal primal-dual pair $(\boldsymbol{w}, \boldsymbol{\alpha})$
Initialization:
$\boldsymbol{w} = \boldsymbol{0}$, $\big(\mathcal{Y}_i = \{y_i\}, i \in \mathcal{I}\big)$, $\left(\alpha_i(y) = \begin{cases} \frac{1}{m} \text{ if } & y = y_i \\ 0 \text{ otherwise} \end{cases}, i \in \mathcal{I}\right)$
$\hat{\boldsymbol{A}} := (\boldsymbol{\psi}_i(y_i) \mid i \in \mathcal{I})$, $\hat{\boldsymbol{b}} := (\Delta_i(y_i) \mid i \in \mathcal{I})$
converged := false, regime := online, $\big(\text{satisfied}(i) := \text{false}, i \in \mathcal{I}\big)$
repeat
    G := 0, num_satisfied := 0, num_checked := 0
    forall the $i \in \mathcal{I}$ do
        if satisfied(i) = false then
            num_checked := num_checked + 1
            $u_1 := \text{argmax}_{y \in \mathcal{Y}} s_i(y, \boldsymbol{w})$
            $u_2 := \text{argmax}_{y \in \mathcal{Y}_i} s_i(y, \boldsymbol{w})$
            if $s_i(u_1, \boldsymbol{w}) > s_i(u_2, \boldsymbol{w})$ then
                $\hat{u} := u_1$
            else
                $\hat{u} := u_2$
            $G_i := s_i(\hat{u}, \boldsymbol{w}) - m \sum_{y \in \mathcal{Y}_i} \alpha_i(y) s_i(y, \boldsymbol{w})$
            $G := G + G_i$
            if $G_i \leq \varepsilon$ then
                satisfied(i) := true
                num_satisfied := num_satisfied + 1
            else
                if $\hat{u} = u_1$ then
                    $\mathcal{Y}_i := \mathcal{Y}_i \cup u_1$
                    $\hat{\boldsymbol{A}} := \hat{\boldsymbol{A}} \cup \boldsymbol{\psi}_i(u_1)$
                    $\hat{\boldsymbol{b}} = \hat{\boldsymbol{b}} \cup \Delta_i(u_1)$
                if regime = online then
                    $\hat{v} := \underset{y \in \{y' \in \mathcal{Y}_i \mid \alpha_i(y') > 0\}}{\text{argmax}} \delta(i, \hat{u}, y)$
                    $\tau := \min\left\{1, \frac{\lambda(s_i(\hat{u}, \boldsymbol{w}) - s_i(\hat{v}, \boldsymbol{w}))}{\alpha_i(\hat{v}) \|\boldsymbol{\psi}_i(\hat{u}) - \boldsymbol{\psi}_i(\hat{v})\|^2}\right\}$
                    $\boldsymbol{w} := \boldsymbol{w} + (\boldsymbol{\psi}_i(\hat{v}) - \boldsymbol{\psi}_i(\hat{u})) \frac{\tau \alpha_i(\hat{v})}{\lambda}$
                    $\alpha_i(\hat{u}) := \alpha_i(\hat{u}) + \tau \alpha_i(\hat{v})$
                    $\alpha_i(\hat{v}) := \alpha_i(\hat{v}) - \tau \alpha_i(\hat{v})$
    if num_checked = $m$ then
        if regime = batch $\wedge$ $G \leq \varepsilon$ then
            converged := true
        if $m \cdot K \leq$ num_satisfied then
            regime := batch
    if $\forall i \in \mathcal{I}$, satisfied(i) = true then
        satisfied(i) := false, $i \in \mathcal{I}$
    if converged = false then
        Update $\hat{\boldsymbol{\alpha}} = (\alpha_i(y) \mid y \in \mathcal{Y}_i, i \in \mathcal{I})$ by solving the reduced problem (12)
until converged = true;
```

in a rare case when $G(\boldsymbol{w}, \boldsymbol{\alpha}) = \varepsilon$, $\forall i \in \mathcal{I}$. We resolve the problem by introducing two optimization regimes: on-line and batch. FASOLE is started in the on-line regime, $\text{regime} = \text{online}$, during which the "for" loop instantly updates the dual variables identified as non-optimal. As soon as it gets close to the optimum it switches from online to the batch regime. In the batch regime, FASOLE uses the "for" loop only to select new non-optimal variables and to simultaneously evaluate the actual duality gap $G(\boldsymbol{w}, \boldsymbol{\alpha})$. The variable update in the batch regime is done solely by solving the reduced problem (12). FASOLE switches from on-line to batch when a large portion of dual variables are found optimal, in particular, if $G_i(\boldsymbol{w}, \boldsymbol{\alpha}) \leq \varepsilon$ holds for $m \cdot K$ variables at least. We used the value $K = 0.9$ in all our experiments.

4 Relation to Existing Methods

In this section we describe relation between the proposed SDA-WSS2 algorithm (and FASOLE, respectively) and other two instances of the generic SDA algorithm 1 that have been recently proposed for solving the SO-SVM dual. We also mention a relation to two-class SVM solvers which use a similar optimization strategy.

Sequential Dual Method for Structural SVMs (SDM) [1] SDM is among the existing solvers the most similar to our approach. SDM is an instance of the generic SDA using the SMO updated rule (7) similarly to the proposed SDA-WSS2. The main difference lies in the strategy for selecting the variables for update. SDM uses so called maximal violating pair (MVP) strategy which returns the variables most violating Karush-Kuhn-Tucker (KKT) conditions of the SO-SVM dual (2). Specifically, it finds $\hat{u}$ by (10), similarly to our approach, however $\hat{v}$ is set to

$$\hat{v} = \operatorname*{argmin}_{y \in \{y' \in \mathcal{Y} \mid \alpha_i(y') > 0\}} s_i(v, \boldsymbol{w}) \, .$$

The MVP strategy can be seen as a cruel approximation of WSS2 strategy. Indeed, MVP maximizes the improvement $\delta(i, u, v)$ if we neglect the terms containing λ and $\|\boldsymbol{\psi}_i(u) - \boldsymbol{\psi}_i(v)\|$ in the formula (9). Note that WSS2 strategy introduces only a negligible computational overhead if compared to the MVP. We show experimentally that the proposed the SDA with WSS2 strategy consistently outperforms the SDM using the MVP strategy as it requires consistently less number of oracle calls. Similar behavior showing that the WSS2 strategy outperforms the MVP strategy has been observed for the two-class SVM solvers in [3,4].

Block-Coordinate Frank-Wolfe (BCFW) [9] BCFW is an instance of the generic SDA constructing the point $\boldsymbol{\beta}$ as

$$\beta_j(y) = \begin{cases} \frac{1}{m} \text{ if } y = u \wedge i = j \\ 0 \text{ if } y \neq u \wedge i = j \\ \alpha_j^{(t)}(y) \text{ if } j \neq i \end{cases} \tag{13}$$

where $\hat{u}$ is selected by (10); we call this variable selection strategy as the BCFW update rule. Unlike the SMO update rule used by SDA-WSS2 and SDM, the BCFW rule changes the whole block of $|\mathcal{Y}|$ variables $\boldsymbol{\alpha}_i$ at once. It can be shown that the BCFW rule selects among the admissible points the one in which direction the derivative of the SO-SVM dual objective is maximal. Hence, the resulting SDA algorithm using the BCFW rule can be seen as the steepest feasible ascent method. Empirically it also behaves similarly to the steepest ascent methods, i.e. it exhibits fast convergence at the first iterations but stalls as it approaches optimum. The slow convergence is compensated by simplicity of the method. Specifically, it can be shown that the BCFW rule admits to express the update rule, and consequently the whole algorithm, without explicitly maintaining the dual variables $\boldsymbol{\alpha}$. That is, the BCFW algorithm operates only with the primal variables though it maximizes the dual SO-SVM objective. The empirical evaluation shows that the BCFW converges significantly slower compared to the SDA-WSS2, as well as SDM, both using the SMO update rule. Similar behavior have been observed when the BCFW update rule is applied to two-class SVM problem [4].

Two-class SVM solvers using the SDA with WSS2 [3][4][5] The SDA methods with WSS2 have been first applied for solving the two-class SVM with L2-hinge loss [4][5] and with L1-hinge loss in [3]. A similar method was also proposed in [6]. The SDA with WSS2 is the core solver of LibSVM [2] being currently the most popular SVM implementation. The main difference to the proposed SDA-WSS2 lies in the form and the size of the quadratic programs these methods optimize. In particular, the two-class SVM dual has only a single linear constraint and m variables. In contrast, the SO-SVM dual has m linear constraints and $m|\mathcal{Y}|$ variables. The extreme size of the SO-SVM does not admit operations used in two-class SVM solvers like maintaining all dual variables and buffering the columns of the Hessian matrix. In addition, selection of the most violated constraint via the classification oracle is expensive in the case of SO-SVM and must be reduced. The proposed method thus introduces a sparse representation of the SO-SVM dual and a set of heuristics to reflect the mentioned difficulties.

In addition, our convergence Theorem 1 provides an upper bound on the number of updates to achieve the ε-optimal solution. To our best knowledge no similar result is known for the two-class SVM solvers. In particular, only asymptotical convergence of the SMO type algorithms have been proved so far [16].

5 Experiments

Compared methods. We compared the proposed solver FASOLE (Algorithm 3) against the SDM [1] and BCFW [9] which are all instances of the generic SDA algorithm 1. In addition, we compare against the the Cutting Plane Algorithm (CPA) [8,18] being the current gold-standard for SO-SVM learning (e.g. implemented in popular StructSVM library). We also refer to [9] which provides a thorough comparison showing that the BCFW consistently outperforms approximate on-line methods including the exponentiated gradient [11] and the stochastic sub-gradient method [14] hence these methods are excluded from our comparison.

Datasets. We used three public benchmarks which fit to the SO-SVM setting. First, we learn a linear multi-class classifier of isolated handwritten digits from the USPS dataset [10]. Second, we learn OCR for a sequence of segmented handwritten letters modeled by the Markov Chain classifier [17]. Third, we learn a detector of landmarks in facial images based on a deformable part models [12]. For USPS and OCR classifiers we use normalized image intensities as dense features. For the LANDMARK detector we use high-dimensional sparse feature descriptors based on the Local Binary Patterns as suggested in [12]. The classification oracle can be solved by enumeration in the case of USPS and by Viterbi algorithm in the case of OCR and LANDMARK. The three applications require different loss function $\Delta(y, y')$ to measure the performance of the structured classifier. Specifically, we used the 0/1-loss (classification loss) for USPS data, Hamming loss normalized to the number of characters the OCR problem, and the loss for LANDMARK data was the absolute average deviation between the estimated and the ground-truth landmark positions measured in percents of the face size. The datasets are summarized in Table 1.

Implementation. The competing methods are implemented in Matlab. CPA, SDM and FASOLE use the same inner loop quadratic programming solver written in C. SDM and FASOLE implement the same framework described by Algorithm 3 but SDM uses the SMO update with MVP selection strategy of [1]. We do not implement different heuristics proposed in [1] in order to measure the effect of different variable selection strategy being the main contribution of our paper. All methods use the same classification oracles. The oracles for OCR and USPS are implemented in Matlab. The oracle for LANDMARK is implemented in C. All methods use the same stopping condition based on monitoring the duality gap. In contrast to FASOLE and SDM, the authors of BCFW do not provide an efficient way to compute the duality gap. Hence we simply evaluate the gap every iteration and stop BCFW when the goal precision is achieved but we DO NOT count the duality gap computation to the convergence speed and thus the wall clock times for BCFW are biased to lower values. The experiments were run on the AMD Opteron CPU 2600 MHz/256GB RAM.

Evaluation. We measure convergence speed in terms of the effective iterations. One effective iteration equals to m oracle calls where m is the number of examples. Note that CPA algorithm requires one effective iteration to compute a single cutting plane. In contrast, one effective iteration of SDM, BCFW and FASOLE corresponds to m updates. The effective iteration is an implementation independent measure of the convergence time which is correlated with the the real CPU time when the oracle calls dominate the other computations. This is the case e.g. for the OCR and LANDMARK where the oracle calls are expensive. We also record the wall clock time because the competing methods have different overheads, e.g. CPA, SDM and FASOLE call an inner QP solver. We run all methods for a range of regularization constants, specifically, $\lambda \in \{10, 1, 0.1, 0.01\}$. We stopped each method when the ε-optimal solution has been achieved. We set the precision parameter to $\varepsilon = 0.001$ for USPS and OCR, and $\varepsilon = 0.1$ for the LANDMARK problem. Note that the target precision ε is given in terms of the risk function which has different units for different application. Specifically, for USPS and OCR the units is the probability (hence $\varepsilon = 0.001$ seems sufficient) while for LANDMARK the units is the percentage (hence $\varepsilon = 0.1$).

Table 1. Parameters of the benchmark datasets used in comparison

dataset	#training examples	#testing examples	#parameters	structure
USPS	7,291	2,007	2,570	flat
OCR	5,512	1,365	4,004	chain
LANDMARK	5,062	3,512	232,476	tree

Table 2. The number of effective iterations and the wall clock time needed to converge to ε-optimal solution for different setting of the regularization constant λ. The time is measured in seconds for USPS and in hours for OCR and LANDMARK. The last two rows correspond to accumulated time/iterations and the speedup achieved by the proposed solver FASOLE (speedup={CPA,BCFW,SDM}/FASOLE). The BCFW method had problems to converge for low values of λ hence we stopped BCFW when it used the same number of effective iterations as CPA (the slowest method which converged). These cases are marked with brackets. The best results, i.e. minimal number of iterations and the shortest time are printed in bold.

USPS

	CPA		BCFW		SDM		FASOLE	
λ	iter	time	iter	time	iter	time	iter	time [s]
1.000	62	**3.6**	18	61.6	**5**	9.5	**5**	9.8
0.100	101	**6.0**	70	214.2	6	11.6	**5**	9.9
0.010	197	**10.9**	(197)	(538.0)	13	39.12	**5**	14.0
0.001	380	**26.7**	(380)	(1,018.8)	24	399.9	**7**	30.5
total	740	**47.3**	665	1,832.6	48	460.2	**22**	64.2
speedup	33.6	0.73	30.2	28.6	2.2	7.1	1	1

OCR

	CPA		BCFW		SDM		FASOLE	
λ	iter	time	iter	time	iter	time	iter	time [h]
1.000	63	0.23	26	0.41	**8**	0.09	9	**0.04**
0.100	111	0.39	89	1.28	**10**	0.16	13	**0.07**
0.010	257	0.91	(257)	(3.55)	20	0.60	**16**	**0.15**
0.001	655	2.31	(655)	(9.47)	49	6.04	**20**	**0.43**
total	1086	3.83	1027	14.70	87	6.89	**58**	**0.70**
speedup	18.7	5.5	17.7	21.0	1.5	9.8	1	1

LANDMARK

	CPA		BCFW		SDM		FASOLE	
λ	iter	time	iter	time	iter	time	iter	time [h]
10.00	93	2.43	4	0.18	8	0.32	**6**	**0.21**
1.00	165	4.71	20	0.78	11	0.40	**8**	**0.28**
0.10	261	7.82	(261)	(8.52)	30	1.42	**15**	**0.55**
0.01	446	12.20	(446)	(12.14)	131	12.30	**39**	**1.79**
total	965	27.25	731	21.62	180	14.42	**68**	**2.83**
speedup	14.2	9.6	10.8	7.6	2.6	5.1	1	1

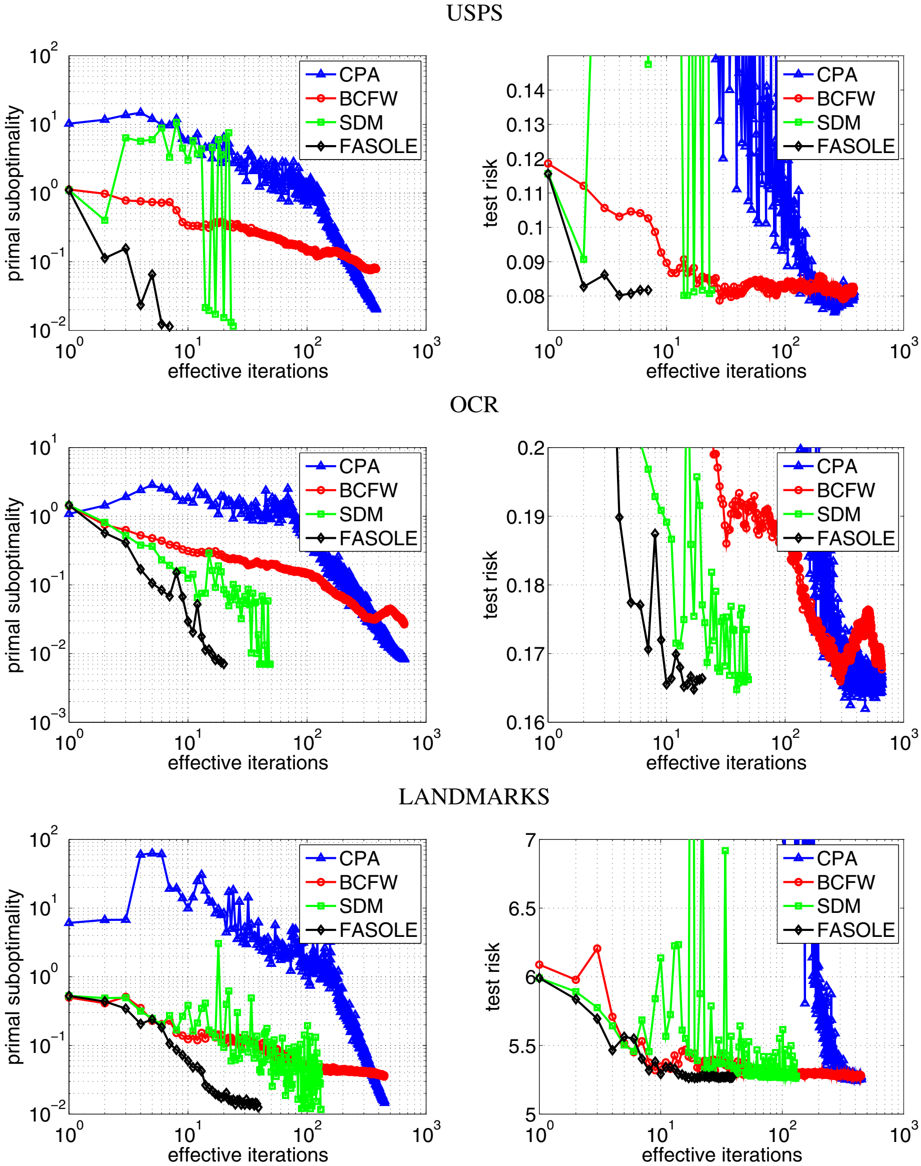

Fig. 1. Convergence curves for the regularization parameter λ which produces the minimal test risk, in particular, $\lambda = 0.01$ for USPS and LANDMARK and $\lambda = 0.001$ for OCR. The left column shows convergence in terms of the primal sub-optimality and the right column convergence of the test risk.

Discussion of the results. Table 2 summarizes the numbers of effective iteration and the wall clock time required by competing methods to achieve the target ε-precision. In sake of space we do not included the final objective because they are almost the same (must not differ more than ε).

The results show that if compared to CPA and BCFW, the proposed FASOLE requires order of magnitude less number of effective iterations on all three datasets. This leads to the speedup in terms of the wall clock time ranging from 5.5 to 21.0 on structured problems OCR and LANDMARKS. The CPA algorithm requires less time on the non-structured USPS benchmark because the risk function of multi-class SVM required by CPA can be evaluated by a single matrix multiplication (very effective in Matlab) unlike the SDA solvers (BCFW, SDM, FASOLE) which compute the loss for individual examples separately (calling function in a loop not effective in Matlab).

Comparison to SDM, the closest method to FASOLE, shows that SDM requires approximately only two times more effective iterations than FASOLE. However, SDM requires much more time in the inner loop, optimizing over buffered features, especially for low values of λ. This results to significantly slower convergence in terms of the wall clock time, specifically, the speedup achieved by FASOLE relatively to SDM ranges from 5.1 to 9.8.

These results show that FASOLE converges to the ε-optimal solution consistently faster on all problems which translate to significant speedup in terms of wall-clock time for the cases where the oracle calls are expensive. The advantage of the FASOLE is especially significant for small values of λ when the speed up is often an order of magnitude. Figure 1 (column 1) shows convergence of the competing methods in terms of the primal sub-optimality $(P(\boldsymbol{w}) - P(\boldsymbol{w}^*))/P(\boldsymbol{w}^*)$, i.e. relative deviation of the primal objective from the optimal value, where $P(\boldsymbol{w}^*)$ was replaced by the maximal achieved dual value. The figures show that FASOLE converges consistently faster from the beginning to the end. This implies that it beats the competing methods for the whole range of the precision parameter ε and not only the particular setting the results of which are reported in Table 2.

Some authors advocate that the optimality in terms of the objective function is not the primal goal and instead they propose to stop the algorithm based on monitoring the test risk. Therefore we also record convergence of the test risk which is presented in the second column of Figure 1. We see that the convergence of the test risk closely resembles the convergence of the objective function (compare the first and the second column).

6 Conclusions

In this paper we have proposed a variant of the sequential dual ascent algorithm for optimization of the SO-SVM dual. The proposed algorithm, called FASOLE, uses working set selection strategy which has been previously used for optimization of simpler QP tasks emerging in learning the two-class SVM. We provide a novel convergence analysis which guarantees that FASOLE finds the ε-optimal solution in $\mathcal{O}(\frac{1}{\varepsilon^2})$ time. The empirical comparison indicates that FASOLE consistently outperforms the existing state-of-the-art solvers for the SO-SVM achieving up to an order of magnitude speedups while obtaining the same precise solution.

Acknowledgment. The author was supported by the project ERC-CZ LL1303.

References

1. Balamurugan, P., Shevade, S.K., Sundararajan, S., Sathiya Keerthi, S.: A sequential dual method for structural SVMs. In: SIAM Conference on Data Mining (2011)
2. Chang, C.-C., Lin, C.-J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011)
3. Fan, R., Chen, P.H., Lin, C.J.: Working set selection using second order information for training support vector machines. JMLR 6, 1889–1918 (2005)
4. Franc, V.: Optimization Algorithms for Kernel Methods. PhD thesis, Center for Machine Perception, K13133 FEE Czech Technical University, Prague, Czech Republic (2005)
5. Franc, V., Hlaváč, V.: Simple solvers for large quadratic programming tasks. In: Kropatsch, W.G., Sablatnig, R., Hanbury, A. (eds.) DAGM 2005. LNCS, vol. 3663, pp. 75–84. Springer, Heidelberg (2005)
6. Glasmachers, T., Igel, C.: Maximum-gain working set selection for SVMs. Journal of Machine Learning Research (2006)
7. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel Methods: Support Vector Machines. MIT Press (1998)
8. Joachims, T., Finley, T., Yu, C.: Cutting-plane training of structural SVMs. Machine Learning 77(1), 27–59 (2009)
9. Lacoste-Julien, S., Jaggi, M., Schmidt, M., Pletscher, P.: Block-coordinate frank-wolfe optimization for structural SVMs. In: ICML (2013)
10. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Handwritten digit recognition with a back-propagation network. In: Lisboa, P.G.J. (ed.) Neural Netwotks, Current Applications. Chappman and Hall (1992)
11. Collins, M., Globerson, A., Koo, T., Carreras, X., Bartlett, P.L.: Exponentiated gradient algorithms for conditional random fileds and max-margin markov networks. JMLR 9, 1775–1822 (2008)
12. Uricar, M., Franc, V., Hlavac, V.: Detector of facial landmarks learned by the structured output SVM. In: VISAPP (2012)
13. Ratliff, N., Bagnell, J.A., Zinkevich, M.: (online)subgradient methods for structured predicetions. In: AISTATS (2007)
14. Shalev-Shwartz, S., Singer, Y., Sebro, N., Cotter, A.: Pegasos: primal estimated sub-gradient solver for svm. In: ICML (2007)
15. Shalev-Shwartz, S., Zhang, T.: Trading accuracy for sparsity in optimization problems with sparsity constraints. SIAM Journal on Optimization 20(6), 2807–2832 (2010)
16. Takahashi, N., Nishi, T.: Rigorous proof of termination of smo algorithm for support vector machines. IEEE Transactions on Neural Networks (2005)
17. Taskar, B., Guestrin, C., Koller, D.: Max-margin markov networks. In: NIPS, vol. 16. MIT Press (2003)
18. Teo, C.H., Vishwanathan, S.V.N., Smola, A.J., Le., Q.V.: Bundle methods for regularized risk minimization. JMLR (2010)
19. Tsochantaridis, I., Joachims, T., Hoffman, T., Altun, Y.: Large margin methods for structured and interdependent output variables. JMLR 6, 1453–1484 (2005)