Bundle Methods for Structured Output Learning — Back to the Roots

Michal Uřičář[1], Vojtěch Franc[1], and Václav Hlaváč[1]

Center for Machine Perception, Department of Cybernetics,
Faculty of Electrical Engineering, Czech Technical University in Prague,
Technická 2, 166 27 Prague 6, Czech Republic
{uricamic, xfrancv, hlavac}@cmp.felk.cvut.cz

Abstract. Discriminative methods for learning structured output classifiers have been gaining popularity in recent years due to their successful applications in fields like computer vision, natural language processing, etc. Learning of the structured output classifiers leads to solving a convex minimization problem, still hard to solve by standard algorithms in real-life settings. A significant effort has been put to development of specialized solvers among which the Bundle Method for Risk Minimization (BMRM) [1] is one of the most successful. The BMRM is a simplified variant of bundle methods well known in the filed of non-smooth optimization. In this paper, we propose two speed-up improvements of the BMRM: i) using the adaptive prox-term known from the original bundle methods, ii) starting optimization from a non-trivial initial solution. We combine both improvements with the multiple cutting plane model approximation [2]. Experiments on real-life data show consistently faster convergence achieving speedup up to factor of 9.7.

Keywords: Structured Output Learning, Bundle Methods, Risk Minimization, Structured Output SVM

1 Introduction

Learning predictors from data is a standard machine learning task. A large number of such tasks are translated into a convex quadratically regularized risk minimization problem

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}\in\mathbb{R}^n} F(\boldsymbol{w}) := \left[\frac{\lambda}{2}\|\boldsymbol{w}\|^2 + R(\boldsymbol{w})\right] . \tag{1}$$

The objective $F\colon \mathbb{R}^n \to \mathbb{R}$, referred to as the regularized risk, is a sum of the quadratic regularization term and a convex empirical risk $R\colon \mathbb{R}^n \to \mathbb{R}$. The scalar $\lambda > 0$ is a predefined constant and $\boldsymbol{w} \in \mathbb{R}^n$ is a parameter vector to be learned. The quadratic regularization term serves as a mean to constraint the space of solutions in order to improve generalization. The empirical risk evaluates a match between the parameters $\boldsymbol{w}$ and training examples. The risk typically splits into a sum of convex functions $r_i\colon \mathbb{R}^n \to \mathbb{R}$, i.e. the risk reads

$$R(\boldsymbol{w}) = \sum_{i=1}^{m} r_i(\boldsymbol{w}) . \tag{2}$$

This paper proposes efficient optimization algorithm for the instances of the learning problem (1) when evaluation of the functions $r_i(\boldsymbol{w})$ and their sub-gradients $\boldsymbol{r}'_i(\boldsymbol{w}) \in \mathbb{R}^n$ is expensive, yet tractable.

In particular, our research was motivated by real-life applications of the Structured Output Support Vector Machine (SO-SVM) classifier, learning of which has been formulated by [3] as follows. Given a training set of examples of input-output pairs $\{x_i, y_i\}_{i=1}^m \in (\mathcal{X} \times \mathcal{Y})^m$, assumed to be i.i.d. from an unknown p.d.f. $P(x, y)$, we want to learn a parameter vector $\boldsymbol{w} \in \mathbb{R}^n$ of a linear classifier

$$h(x; \boldsymbol{w}) = \arg\max_{y \in \mathcal{Y}} \langle \boldsymbol{w}, \boldsymbol{\Psi}(x, y) \rangle \,, \tag{3}$$

where $\boldsymbol{\Psi} \colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^n$ is a fixed mapping from the input-output space onto the space of parameters. The ultimate goal is to find the parameters $\boldsymbol{w}$ which minimize the expected risk $E_{p(x,y)}[\ell(y, h(x; \boldsymbol{w}))]$ for a given loss function $\ell \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. The problem (1) was shown to be a good proxy for the minimization of the expected risk, which is not possible due to unknown $P(x, y)$. The function $r_i(\boldsymbol{w})$ is set to be a convex approximation of the loss $\ell(y_i, h(x_i; \boldsymbol{w}))$ which reads

$$r_i(\boldsymbol{w}) = \max_{y \in \mathcal{Y}} \left[\ell(y_i, y) + \left\langle \boldsymbol{w}, \boldsymbol{\Psi}(x_i, y) - \boldsymbol{\Psi}(x_i, y_i) \right\rangle \right] . \tag{4}$$

By Danskin's theorem [4, Proposition B.25] the sub-gradient of $r_i(\boldsymbol{w})$ can be computed as

$$\boldsymbol{r}'_i(\boldsymbol{w}) = \boldsymbol{\Psi}(x_i, \hat{y}_i) - \boldsymbol{\Psi}(x_i, y_i) \,, \tag{5}$$

where $\hat{y}_i = \arg\max_{y \in \mathcal{Y}} [\ell(y_i, y) + \langle \boldsymbol{w}, \boldsymbol{\Psi}(x_i, y) \rangle]$. Evaluation of $r_i(\boldsymbol{w})$ as well as $\boldsymbol{r}'_i(\boldsymbol{w})$ is often expensive due to the huge size of the output set $\mathcal{Y}$, e.g. $\mathcal{Y}$ can be a set of all segmentations of an input image $x \in \mathcal{X}$. We refer to the problem (1) with the risk $R(\boldsymbol{w})$ defined by (4) as the SO-SVM learning problem. It can be transformed to the equivalent quadratic program (QP) with the number of constraints linearly proportional to the number of classifier outputs $|\mathcal{Y}|$. Hence, using existing off-the-shelf solvers is not feasible.

Currently used approaches for the SO-SVM learning involve *approximative on-line algorithms* and *precise methods*. The approximative methods are often fast, especially at early optimization stages, but have no clear stopping condition. Moreover, they require educated setting of the learning rate and are sensitive to improperly scaled data. The prominent representatives of the approximative methods are variants of the Stochastic Gradient Descent (SGD) algorithm [5, 6].

The precise methods are slower but provide theoretically grounded stopping condition based on the optimality certificate. Currently the most popular precise solver is the Bundle Method for Risk Minimization (BMRM) proposed by [1]. It can be readily applied to an arbitrary instance of the problem (1) requiring only an oracle which evaluates the risk (4) and its sub-gradient (5). The BMRM has been proved to converge to ε-optimal solution after $\mathcal{O}(\frac{1}{\varepsilon})$ iterations at most. [7] proposed a variant of the column generation algorithm solving "1-slack" reformulation of the QP equivalent to the SO-SVM learning problem. The solver was implemented in the Though discovered independently, the StructSVM solver is exactly the same as the instance of the BMRM for the SO-SVM learning.

The BMRM is a simplified variant of bundle methods (BM), which are standard tools in non-smooth optimization [8]. In this paper, we propose two improvements of the BMRM which significantly speed up its convergence. The first improvement is the usage of adaptive prox-term, known from the original BM, which has a significant stabilization effect on the convergence. We also propose a novel strategy for setting the strength of the prox-term which is suitable for the problem (1). The second improvement is to start optimization from a non-trivial initial solution. This improvement cannot be used without the first one due to the lack of non-trivial (i.e. not centered at zero) prox-term like in the original BMRM. In addition, we combined these two improvements with the multiple cutting plane risk approximation [2].

The paper is organized as follows. In Section 2 we outline the standard BM, their relation to the BMRM and the source of inefficiency of the BMRM. Section 3 describes the proposed improvements. Experimental evaluation is given in Section 4 and Section 5 concludes the paper.

2 Existing methods

Let us assume for a moment a special variant of the problem (1) without the regularization term, i.e. $\lambda = 0$, then the problem becomes

$$\boldsymbol{w}^* \in \arg\min_{\boldsymbol{w}\in\mathbb{R}^n} R(\boldsymbol{w})\,. \tag{6}$$

Thanks to its convexity, the risk $R(\boldsymbol{w})$ can be approximated by its cutting plane (CP) model

$$R_t(\boldsymbol{w}) = \max_{i=1,\ldots,t} \left[R(\boldsymbol{w}_i) + \langle R'(\boldsymbol{w}_i), \boldsymbol{w} - \boldsymbol{w}_i\rangle\right], \tag{7}$$

where $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t$ are the points at which the risk $R(\boldsymbol{w})$ is sampled and $R'(\boldsymbol{w}_i) \in \mathbb{R}^n$, $i = 1, \ldots, t$, denotes sub-gradients computed at these points. The CP model $R_t(\boldsymbol{w})$ is a piece-wise linear under-estimator of the risk $R(\boldsymbol{w})$ which is tight at the points $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t$. Figure 1 illustrates the CP approximation on a simple function.

The cutting plane algorithm [9] is a simple iterative procedure which exploits the CP model to solve the problem (6). Starting from an initial solution $\boldsymbol{w}_1 \in \mathbb{R}^n$, the CP algorithm computes the new iterates by solving the *reduced problem* $\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^n} R_t(\boldsymbol{w})$. The iterations generated by the CP algorithm show a strong zig-zag behavior, especially at the early iterations when the CP model is inaccurate, causing it to converge slowly.

The BM [10] refine the CP algorithm by adding a quadratic prox-term to the reduced problem, i.e. the next iterate becomes $\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^n} \left[R_t(\boldsymbol{w}) + \alpha_i \|\boldsymbol{w} - \boldsymbol{w}_i^+\|^2\right]$. where $\boldsymbol{w}_i^+$ is the prox-center and α_i is the prox-term penalty parameter. If the improvement in the objective value is sufficiently large, i.e. if $R(\boldsymbol{w}_t) - R(\boldsymbol{w}_{t+1}) \geq \gamma_t$ holds, the prox-center is updated to $\boldsymbol{w}_{t+1}^+ = \boldsymbol{w}_{t+1}$. Otherwise, the prox-center is unchanged $\boldsymbol{w}_{t+1}^+ = \boldsymbol{w}_t^+$. The prox-term reduces influence of the inaccurate CP model by constraining the distance between consecutive iterations, thereby removing the detrimental zig-zag behavior of the CP algorithm. The BM is controlled by two rules, with significant impact on the convergence [8]. The first rule defines the minimal decrease threshold γ_t and, the second rule sets the prox-term penalty α_t.

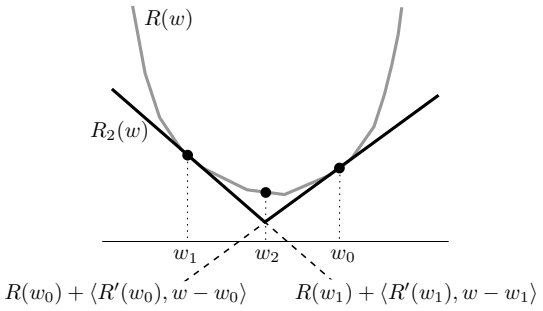


Fig. 1. A convex function $R(\boldsymbol{w})$ can be approximated by a collection of linear under-estimators (cutting planes).

[1] adopted the original BM for the specific problem (1) objective of which already contains a quadratic term. In particular, [1] propose to replace the problem (1) by the following *reduced problem*

$$\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^n} F_t(\boldsymbol{w}) := \left[\frac{\lambda}{2}\|\boldsymbol{w}\|^2 + R_t(\boldsymbol{w})\right] . \tag{8}$$

The reduced problem objective $F_t(\boldsymbol{w})$ is obtained from (1) by replacing the risk $R(\boldsymbol{w})$ with its CP model $R_t(\boldsymbol{w})$ while the quadratic regularization term is unchanged. The regularization term serves as a natural prox-center. This is an elegant solution that avoids designing rules for updating the prox-center penalty and the sufficient decrease threshold which are needed in the standard BM.

Starting from an initial guess $\boldsymbol{w}_1 \in \mathbb{R}^n$, the BMRM iteratively solves the reduced problem (8) and uses the new iterate $\boldsymbol{w}_{t+1}$ to update the CP model (7) which becomes progressively more accurate. This process is repeated until a gap between the primal and the reduced objective gets below a prescribed $\varepsilon > 0$.

The prox-term penalty is in the BMRM replaced by a fixed regularization parameter λ and the prox-center is constantly zero . For low values of λ, the influence of the quadratic term is weak and the BMRM becomes closer to the CP algorithm, i.e. the BMRM again exhibits a zig-zag behavior and, consequently, a slow convergence. The detrimental effect of a low λ is also seen from the upper bound on the maximal number of iterations $\mathcal{O}(\log_2 \lambda + \frac{C}{\lambda\epsilon})$ derived in [1]. The mentioned inefficiency of the BMRM can have serious practical implications because the optimal value of λ is unknown and thus one needs to train with the whole range of λ's including low values which can require prohibitively many iterations and thus long computational times.

3 Improved BMRM

In this section, we propose two improvements of the original BMRM which speed up its convergence. First, we propose to use the prox-term in the definition of the reduced

Algorithm 1 Prox-BMRM

Require: $\varepsilon > 0, T > 0, K > 0$, $\boldsymbol{w}_1 \in \mathbb{R}^n$

1: Set $\alpha_1 = 0$ and $\gamma_t = \infty$
2: **repeat**
3: Solve the reduced problem $\boldsymbol{w}_{t+1}^{\alpha_t} = \arg\min_{\boldsymbol{w} \in \mathbb{R}^n} F_t(\boldsymbol{w}, \alpha_t)$
4: **if** $F(\boldsymbol{w}_t) - F(\boldsymbol{w}_{t+1}^{\alpha_t}) \geq \gamma_t$ **then**
5: accept the solution and set: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_{t+1}^{\alpha_t}, \quad \alpha_{t+1} = \alpha_t, \quad \gamma_{t+1} = \gamma_t$
6: **else**
7: Find the minimal $\hat{\alpha} \in \{2^i\}_0^\infty$ such that $\|\boldsymbol{w}_{t+1}^{\hat{\alpha}} - \boldsymbol{w}_t\| \leq K$, where $\boldsymbol{w}_{t+1}^{\alpha} = \arg\min_{\boldsymbol{w} \in \mathbb{R}^n} F_t(\boldsymbol{w}, \alpha)$
8: Set $\boldsymbol{w}_{t+1} = \boldsymbol{w}_{t+1}^{\hat{\alpha}}$, $\alpha_{t+1} = \hat{\alpha}$ and $\gamma_{t+1} = \frac{F(\boldsymbol{w}_{t+1}^{\hat{\alpha}})}{T} - \frac{F_t(\boldsymbol{w}_{t+1}^0)}{T(1-\varepsilon)}$
9: **end if**
10: **until** $F(\boldsymbol{w}_{t+1}) - F_t(\boldsymbol{w}_{t+1}^0) \leq \varepsilon \cdot |F(\boldsymbol{w}_{t+1})|$

problem in order to avoid the zig-zag behavior. The strength of the prox-term is adaptively adjusted by a novel strategy which we derived for the problem (1). Second, we propose to start the optimization from a non-trivial solution. Both of these improvements can be readily combined with the multiple cutting plane model which we proposed in [2]. The next sections detail the improvements.

3.1 Prox-BMRM

As the first improvement, we propose to integrate a quadratic prox-term to the objective of the reduced problem in order to prevent the zig-zag behavior of the BMRM. This modification, which we call Prox-BMRM, returns the BMRM algorithm closer to its roots, i.e. to the original BM. The difference when compared to the classical BM is that i) we do not approximate the original quadratic regularizer by the CP model and ii) we propose new rules for adjusting the prox-term penalty and the minimal improvement threshold.

The reason behind the additional prox-term is to prevent overly big changes of the solution in two consecutive iterations. To this end, we require that the Euclidean distance between two consecutive iterations $\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|$ is not larger than some reasonably chosen constant $K > 0$. This constraint is implemented by adding a prox-term to the objective function of the reduced problem, i.e. the modified objective becomes

$$F_t(\boldsymbol{w}, \alpha) := \left[\frac{\lambda}{2} \|\boldsymbol{w}\|^2 + R_t(\boldsymbol{w}) + \alpha \|\boldsymbol{w} - \boldsymbol{w}_t\|^2 \right] , \tag{9}$$

where $\alpha \geq 0$ is the prox-term penalty. Similarly to the original BMRM, the Prox-BMRM computes the new iterate by minimizing the reduced objective (9) with the value of α set adaptively. The optimization schema is described in Algorithm 1.

In each iteration, the Prox-BMRM first tries to compute new iterate by minimizing (9) with the prox-center penalty α used in the previous step (line 3). If the new iteration sufficiently improves the primal objective, i.e. its value decreases by more than γ_t (line 4), the solution is accepted and the setting of the penalty α_t as well as the minimal improvement threshold γ_t are unchanged. If the improvement is not sufficient the

prox-term penalty is tuned to guarantee that the distance between the previous and the new iterate is not higher than the constant K (line 7). At the same time, the minimal improvement threshold is set to a new value γ_{t+1}. It is easy to show that if the improvement in all following iterations is not less than γ_{t+1} (i.e. condition on line 4 holds) then the stopping condition is satisfied after at most T iterations. In turn, the prox-center penalty is readjusted not later than after T iterations. To sum up, the Prox-BMRM guarantees in each iteration that either the primal objective is sufficiently improved or the new iterate is not overly far from the previous one. We will experimentally show that this strategy avoids the zig-zag behavior and also significantly decreases the number of iterations needed to converge to the ε-optimal solution.

Compared to the original BMRM, the proposed Prox-BMRM introduces an additional overhead because the solution of the reduced problem can be required several times in a single iteration. The overhead is not dramatic, moreover, it can be significantly reduced by using several tricks. First, in the search for α on line 7 one should use the fact that $\|\boldsymbol{w}_{t+1}^{\alpha_1} - \boldsymbol{w}_t\| > \|\boldsymbol{w}_{t+1}^{\alpha_2} - \boldsymbol{w}_t\|$ holds for any $\alpha_1 < \alpha_2$ which follows from the strict-convexity of the quadratic prox-term. In addition, the search can start from the previous value α_t instead of always going sequentially form $\alpha = 0$. Second, one can significantly speed up solving the reduced problem by using the warm start strategy. Third, the stopping condition on line 10, which also requires solving the reduced problem with $\alpha = 0$ to get lower bound on the optimum, does not need to be evaluated in every iteration. It turns out to be sufficient to evaluate the stopping condition only when the α readjusting takes place. With these tricks implemented, we observed that the reduced problem is solved on average 2-3 times instead of 1 times (like in the original BMRM) which constitutes a negligible increase of computation time. This increase is amply compensated by the reduced number of the iterations.

Besides the precision parameter ε, the Prox-BMRM algorithm requires setting of the initial solution $\boldsymbol{w}_1$ and two constants: K which is the maximal distance between two consecutive iterations and T which is the maximal number of iterations without readjusting the prox-center penalty α. A way to find a non-trivial initial solution, i.e. $\|\boldsymbol{w}_1\| > 0$, is discussed in the next section. We found that setting $T = 100$ and $K = 0.01\|\boldsymbol{w}_1\|$ worked consistently well in all our experiments.

Algorithm 1 reduces the solution of (1) to a sequence of problems (9). The problem (9) is equivalent to QP: $\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^n} \left[\frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \alpha_t\|\boldsymbol{w} - \boldsymbol{w}_t\|^2 + \xi\right]$, s. t. $\xi \geq R(\boldsymbol{w}_i) + \langle R'(\boldsymbol{w}_i), \boldsymbol{w} - \boldsymbol{w}_i\rangle,\ i = 0, \ldots, t-1$. In practice, the number of cutting planes t required by Algorithm 1 to converge is usually much lower than the dimension n of the parameter vector $\boldsymbol{w} \in \mathbb{R}^n$. Thus one can benefit from solving the reduced problem (9) in its dual formulation, which form is very similar to the one used in the standard BMRM.

3.2 Initialization by on-line methods (warm start)

In contrast to the original BMRM, it is reasonable to start the proposed Prox-BMRM from a non-trivial solution, which is possible thanks to the prox-term added to the objective. We use the following two strategies to find the initial solution.

The first strategy finds the initial solution by computing a few iterations of an on-line algorithm. In particular, we use a variant of the SGD [5]. We run 10 passes of

the SGD and keep track of the minimal value of the $F(\boldsymbol{w})$. Finally, we take the SGD solution, which minimizes $F(\boldsymbol{w})$ as the initial point of the Prox-BMRM algorithm.

The second strategy is based on reusing previously obtained solutions. For example, during validation stage, one typically needs to train the classifier with several values of the regularization constant $\lambda \in \Lambda$, where $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. Since the BMRM converges quickly with higher values of λ, it is straightforward to exploit the ordering on Λ, e.g. $\lambda_1 > \lambda_2 > \cdots > \lambda_n$. With such constellation, it is obvious that one can use the solution vector $\boldsymbol{w}^{\lambda_i}$ as an initial solution for computing $\boldsymbol{w}^{\lambda_{i+1}}$.

4 Experiments

In this section, we demonstrate experimentally the effect of the proposed improvements. We use three different instances of the SO-SVM learning as benchmarks. We first briefly describe the three benchmarks and then we present the evaluation.

4.1 OCR

We consider the optical character recognition (OCR) problem as the first benchmark. We use the MNIST database[1] composed of labeled examples of handwritten numerals. The classifier input x is a gray scale image 28×28 pixels large. The classifier output y is a digit name, i.e. $y \in \mathcal{Y} = \{0, \ldots, 9\}$. We model each class by a single template image $w_y \in \mathbb{R}^{28\times 28}$, $y \in \mathcal{Y}$. As the scoring function of the classifier (3) we use $\langle \boldsymbol{w}, \boldsymbol{\Psi}(x, y)\rangle = \langle x, \boldsymbol{w}_y\rangle$. The parameter vector $\boldsymbol{w} \in \mathbb{R}^n$ has dimension $n = 7,840$ resulting from a column-wise concatenation of 10 templates $\boldsymbol{w}_y$, $y \in \mathcal{Y}$, We use the standard classification 0/1-loss defined to be $\ell(y, y') = 1$ for $y \neq y'$ and $\ell(y, y') = 0$ otherwise. With these definitions, the classifier (3) becomes an instance of a linear multi-class SVM classifier.

We train on all $m = 60,000$ training examples.

4.2 Facial landmark detection

We consider learning of a facial landmark detector as the second benchmark. We follow the approach of [11], where the landmark detection is posed as an instance of the SO-SVM classifier (3). The classifier input $x \in \mathcal{X}$ is 40×40 image containing a face. The classifier outputs $y = (y^1, \ldots, y^L) \in \mathcal{Y} = \mathcal{N}^{2\times L}$ being a set of 2D coordinates of L landmarks like corners of the eyes, etc. Evaluation of the classifier (3) leads to solving an instance of dynamic programing (DP). The loss function measures the mean deviation between the ground truth landmark positions y and their estimates y', i.e. $\ell(y, y') = \kappa(y)\frac{1}{L}\sum_{j=0}^{L-1} \|y^j - y'^j\|$, where $\kappa(s)$ is a normalization constant ensuring that the loss is scale invariant.

We trained the landmark detector on a set of $m = 6,919$ images with manually annotated landmark positions. In our experiment, $L = 8$ and the dimensionality of $\boldsymbol{w} \in \mathbb{R}^n$ was $n = 232,476$.

[1] http://yann.lecun.com/exdb/mnist/

4.3 Number plate segmentation

We consider segmentation of car number plate images as the third benchmark. The classifier input $x \in \mathcal{X}$ is an image $H \times W$ pixels large which contains a number plate, i.e. a line of text composed of a known set of characters. The columns of the input image x are features extracted from intensity values of a corresponding column of a raw image taken by a camera. The classifier outputs image segmentation $y = (s_1, \ldots, s_L) \in Y$ where $s = (a, k)$, $a \in A$ is a character code and $k \in \{1, \ldots, W\}$ is a character position. An admissible segmentation $y \in Y$ must satisfy

$$k(s_1) = 1\,, W = k(s_L) + \omega(s_L) - 1\,, \quad k(s_i) = k(s_{i-1}) + \omega(s_{i-1})\,, \forall i > 1\,, \quad (10)$$

where $\omega\colon A \to \mathcal{N}$ are widths of the characters. The constraints (10) guarantee that the segmentation y covers the whole image x by a sequence of characters $a_1, \ldots, a_L$ which do not overlap. Each character $a \in A$ is modeled by a template image $\nu_a \in \mathbb{R}^{H \times \omega(a)}$. The parameter vector $\boldsymbol{w} \in \mathbb{R}^n$ to be learned is a column-wise concatenation of all templates ν_a, $a \in A$. The scoring function of the classifier (3) computes the correlation between the image x and the character templates placed one by one according to the segmentation $y \in Y$, i.e. $\langle \boldsymbol{\Psi}(x, y), \boldsymbol{w} \rangle = \sum_{i=1}^{L(y)} \sum_{j=1}^{\omega(a(s_i))} \langle \mathrm{col}(x, j + k(s_i) - 1), \mathrm{col}(\boldsymbol{w}_{a(s_i)}, j) \rangle$, where $\mathrm{col}(I, i)$ denotes i-th column of the image I. The loss function measures the number of incorrectly segmented columns w.r.t. to the annotated segmentation. Evaluation of the classifier (3), as well as evaluation of $r_i(\boldsymbol{w})$ and its sub-gradient $\boldsymbol{r}_i(\boldsymbol{w})$, leads to an instance of DP.

We used $m = 6,788$ annotated images for training. The parameter vector $\boldsymbol{w}$ had $n = 4,059$ components.

4.4 Results

We compare the original BMRM with the proposed Prox-BMRM described by Algorithm 1. In addition, we combine the Prox-BMRM with the multiple cutting model approximation of the risk [2]. We use Prox-P-BMRM to denote Algorithm 1 with P cutting plane models.

As for the second improvement, we have experimented with two different strategies providing initial solution to the Prox-(P)-BMRM. In Benchmark 1, the strategy of reusing the previous solutions obtained in validation of regularization constant was used. In Benchmarks 2 and 3, the initial solution was obtained in a pre-training with 10 iterations of the SGD [5].

We learned the SO-SVM classifier for a range of regularization parameters λ on the training examples. We used exactly the same stopping condition for all the tested algorithms. As a result, all tested algorithms provide the same precise classifier but they require different time to converge. Hence we do not report classification accuracies but only the training time measured in terms of i) the number of iterations and ii) the wall clock time. The obtained results are summarized in Table 1. We see that the Prox-BMRM significantly decreases both the number of iterations and the wall clock time compared to the original BMRM. As expected, the speedup is higher for lower λ's. The speedup is further improved after increasing the number of CP models to $P = 16$. The

Table 1. The number of iterations, the wall clock time in hours and the obtained speedups for all benchmark problems, all tested algorithms and different values of λ.

Benchmark 1: OCR — MNIST (SHOGUN [12] implementation)

Initial solution	$\lambda_1 = 1000$			$\lambda_2 = 100$			$\lambda_3 = 10$			$\lambda_4 = 1$		
REUSE	iter	time	spdup	iter	time	spdup	iter	time	spdup	iter	time	spdup
BMRM	157	0.04878	1	417	0.12836	1	1429	0.45233	1	5932	2.01847	1
Prox-BMRM	226	0.06893	0.7	232	0.07184	1.8	317	0.09939	4.6	698	0.26470	7.6
Prox-P=16-BMRM	244	0.07838	0.6	256	0.08580	1.5	291	0.10703	4.2	408	0.20904	9.7

Benchmark 2: Facial landmark detection (MATLAB implementation)

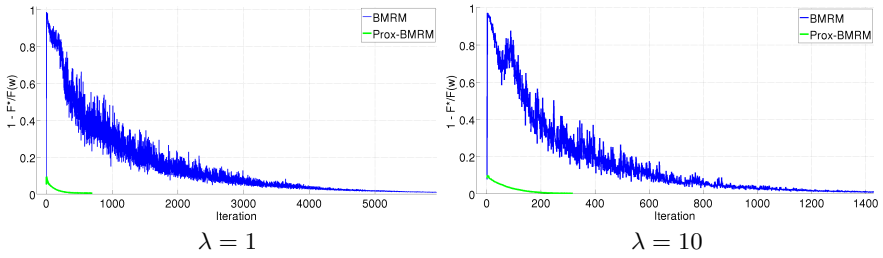
Initial solution	$\lambda_1 = 10000$			$\lambda_2 = 1000$			$\lambda_3 = 100$			$\lambda_4 = 10$		
SGD	iter	time	spdup	iter	time	spdup	iter	time	spdup	iter	time	spdup
BMRM	108	2.492	1	207	8.263	1	442	9.798	1	1084	47.767	1
Prox-BMRM	28	0.693	3.6	61	1.539	5.3	180	4.654	2.1	783	19.772	2.4
Prox-P=16-BMRM	32	0.783	3.2	55	1.301	6.3	142	3.365	2.9	555	14.411	3.3

Benchmark 3: License plate recognition (MATLAB implementation)

Initial solution	$\lambda_1 = 10^5$			$\lambda_2 = 10^4$			$\lambda_3 = 10^3$			$\lambda_4 = 10^2$		
SGD	iter	time	spdup	iter	time	spdup	iter	time	spdup	iter	time	spdup
BMRM	33	0.605	1	87	1.581	1	251	3.923	1	840	13.726	1
Prox-BMRM	34	0.631	1.0	67	1.236	1.3	126	2.065	1.9	286	4.665	2.9
Prox-P=16-BMRM	22	0.404	1.5	37	0.674	2.3	64	0.981	4.0	131	2.444	5.6

improvement is best seen on the license plate benchmark where the parameters have relatively low dimension. Using more CP models is less beneficial on the facial landmark problem where the data are high dimensional and sparse. The maximal speedup 9.7 was obtained on the OCR problem for the lowest λ and Prox-16-BMRM.

To show the effect of the improvements due to the added prox-term we plot convergence curves for the BMRM and the Prox-BMRM on the OCR benchmark in Figure 2. As expected, the convergence curve of the Prox-BMRM is much smoother compared to the BMRM whose curves fluctuate strongly.



$\lambda = 1$ $\lambda = 10$

Fig. 2. Convergence curves of the BMRM and Prox-BMRM on OCR benchmark.

5 Conclusions

In this paper, we have analyzed a source of inefficiency of the BMRM and propose two improvements, which consistently speedup its convergence. The first improvement

brings the BMRM back to the classical BM, from which it has been originally derived. In particular, we propose to use a an adaptive quadratic prox-center to compensate imprecision of the cutting plane model. In addition, the prox-center enables to start from a non-trivial initial solution which can be either found by an imprecise on-line algorithm like the SGD or one can reuse previous solution obtained, e.g. during model selection. We also measured the benefit obtained from combining the proposed improvements with the multiple cutting plane model [2]. We evaluate the proposed improvements on the MNIST data and two real-life applications of the SO-SVM classifiers. The experiments show that the BMRM using the proposed improvements converges consistently faster achieving speedup up to a factor of 9.7.

The proposed algorithm has become a core SO-SVM solver of the SHOGUN Machine Learning Toolbox [12].

Acknowledgements

MU and VH were supported by The Technology Agency of the Czech Republic under Project TE01020197, VF by the Grant Agency of the Czech Republic under Project P202/12/2071 and by EC project FP7-ICT-247525 HUMAVIPS and VH by EC project FP7-288553 CLOPEMA.

References

1. Teo, C., Vishwanathan, S., Smola, A., Quoc, V.: Bundle Methods for Regularized Risk Minimization. Journal of Machine Learning Research **11** (2010) 311–365
2. Uřičář, M., Franc, V.: Efficient Algorithm for Regularized Risk Minimization. In: CVWW '12: Proceedings of the 17th Computer Vision Winter Workshop. (February 2012) 57–64
3. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large Margin Methods for Structured and Interdependent Output Variables. Journal of Machine Learning Research **6** (2005) 1453–1484
4. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific, Belmont, MA (1999)
5. Bordes, A., Bottou, L., Gallinari, P.: SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent. Journal of Machine Learning Research **10** (2009) 1737–1754
6. Shwartz, S., Singer, Y., Srebro, N.: Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In: Proceedings of International Conference on Machine Learning (ICML), ACM Press (2007) 807 – 814
7. Joachims, T., Finley, T., Yu, C.N.: Cutting-Plane Training of Structural SVMs. Machine Learning **77**(1) (2009) 27–59
8. Lemaréchal, C., Nemirovskii, A., Nesterov, Y.: New variants of bundle methods. Mathematical Programming **69** (1995) 111–147
9. Cheney, E., Goldstain, A.: Newton's method for convex programming and Tchebytcheff approximation. Numerische Mathematick **1** (1959) 253–268
10. Lemaréchal, C.: Nonsmooth optimization and descend methods. Technical report, IIASA, Laxenburg, Austria (1978)
11. Uřičář, M., Franc, V., Hlaváč, V.: Detector of facial landmarks learned by the structured output svm. In: VISAPP (1), SciTePress (2012) 547–556
12. Sonnenburg, S., Rätsch, G., Henschel, S., Widmer, C., Behr, J., Zien, A., Bona, F.d., Binder, A., Gehl, C., Franc, V.: The shogun machine learning toolbox. J. Mach. Learn. Res. **99** (August 2010) 1799–1802