Bundle Method for Structured Output Learning

Michal Uřičář
Vojtěch Franc

{uricamic, xfrancv}@cmp.felk.cvut.cz

CTU–CMP–2012–20
Published also as: K333–46/12

September 6, 2012

Available at
ftp://cmp.felk.cvut.cz/pub/cmp/articles/uricar/Uricar-TR-2012-20.pdf

The authors were supported by EC projects FP7-ICT-247525 HUMAVIPS, The Grant Agency of the CTU Project SGS12/187/OHK3/3T/13 and by The Technology Agency of the Czech Republic project TE01020197. Any opinions expressed in this paper do not necessarily reflect the views of the European Community. The Community is not liable for any use that may be made of the information contained herein.

Research Reports of CMP, Czech Technical University in Prague, No. 20, 2012

Published by

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

Bundle Method for Structured Output Learning

Michal Uřičář Vojtěch Franc

September 6, 2012

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 or bio-informatics. Learning of the structured output classifiers leads to solving a convex minimization problem which is not tractable by standard algorithms. A significant effort has been put to development of specialized solvers among which the Bundle Method for Risk Minimization (BMRM) [Teo et al., 2010] is one of the most successful. The BMRM is a simplified variant of bundle methods well known in the filed of non-smooth optimization. The simplicity of the BMRM is compensated by its reduced efficiency. In this paper, we propose several improvements of the BMRM which significantly speeds up its convergence. The improvements involve i) using the prox-term known from the original bundle methods, ii) starting optimization from a non-trivial initial solution and iii) using multiple cutting plane model to refine the risk approximation. Experiments on real-life data show that the improved BMRM converges significantly faster achieving speedup up to a factor of 10 compared to the original BMRM. The proposed method has become a part of the SHOGUN Machine Learning Toolbox [Sonnenburg et al., 2010].

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 pre-defined 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 is often given as a sum of convex functions $r_i \colon \mathbb{R}^n \to \mathbb{R}$, i.e., the risk reads

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

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 [Tsochantaridis et al., 2005] as follows. Given a training set of examples of input-output pairs $\{(x_1, y_1), \ldots, (x_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$, 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

$$h(x; \boldsymbol{w}) = \arg\max_{y \in \mathcal{Y}} \langle \boldsymbol{w}, \boldsymbol{\Psi}(x, y) \rangle \; , \tag{2}$$

where $\boldsymbol{\Psi} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^n$ is a fixed mapping from the input-output space onto the space of parameters. The ultimate goal is to find the paramaters $\boldsymbol{w}$ 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

$$r_i(\boldsymbol{w}) = \max_{y \in \mathcal{Y}} \left[\ell(y_i, y) + \langle \boldsymbol{w}, \boldsymbol{\Psi}(x_i, y) - \boldsymbol{\Psi}(x_i, y_i) \rangle \right] \; . \tag{3}$$

By Danskin's theorem the sub-gradient of $r_i(\boldsymbol{w})$ can be computed as $\boldsymbol{r}'_i(\boldsymbol{w}) = \boldsymbol{\Psi}(x_i, \hat{y}_i) - \boldsymbol{\Psi}(x_i, y_i)$, where

$$\hat{y}_i = \arg\max_{y \in \mathcal{Y}} [\ell(y_i, y) + \langle \boldsymbol{w}, \boldsymbol{\Psi}(x_i, y) \rangle] \; . \tag{4}$$

Evaluation of $r_i(\boldsymbol{w})$ as well as $\boldsymbol{r}'_i(\boldsymbol{w})$ is often expensive due to 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 (3) as the SO-SVM learning problem. The SO-SVM learning problem can be transformed to the equivalent quadratic program with the number of constraints linearly proportional to the number of classifier outputs $|\mathcal{Y}|$. Hence, using exisitng 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, the approximative methods require educated setting of the learning rate and they are sensitive to improperly scaled data. The prominent representatives of the approximative methods are variants of the Stochastic Gradient Descent algorithm [Bordes et al., 2009, Shwartz et al., 2007].

The precise methods, on the other hand, 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 [Teo et al., 2010]. The strongest sides of the BMRM are its simplicity, generality and modularity. The BMRM can be readily applied to an arbitrary instance of the problem (1) requiring only an oracle which evaluates the risk and its sub-gradient. That is, to use it for the SO-SVM learning one only needs to implement evaluation of (3) and (4). The BMRM has been proved to converge at most in $\mathcal{O}(\frac{1}{\varepsilon})$ iterations. [Joachims et al., 2009] proposed a variant of the column generation algorithm solving "1-slack" reformulation of the quadratic program equivalent to the SO-SVM learning problem. The solver was implemented in the popular StructSVM package[1] for several instances of SO-SVM classifier including the Hidden Markov Models or Context Free Grammars. 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 which are standard tools in non-smooth optimization [Lemaréchal et al., 1995]. The simplicity of the BMRM is compensated by its reduced efficiency. In this paper, we propose several improvements of the BMRM which significantly speeds up its convergence. The first improvement is in using the prox-term, known from the original bundle methods, which has a significant stabilizing effect on the convergence. The second improvement is to start optimization not from a scratch, but to enforce the use of e.g. previous solution from the validation cycle or a solution found by a few passes of an approximate on-line algorithm. This improvement cannot be used without the first one, due to the lack of prox-term in original BMRM. The third improvement, independent to the first two, proposes usage of a multiple cutting plane model for the risk instead of a single cutting plane model as in the original BMRM. The multiple cutting plane model offers finer approximation without increasing the computational time required to computed the model.

The paper is organized as follows. In Section 2 we outline the standard bundle methods, 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 Bundle 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

$$\boldsymbol{w}^* \in \arg\min_{\boldsymbol{w}\in\mathbb{R}^n} R(\boldsymbol{w}) \,. \tag{5}$$

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

$$R_t(\boldsymbol{w}) = \max_{i=1,\ldots,t} \left[R(\boldsymbol{w}_i) + \langle R'(\boldsymbol{w}_i), \boldsymbol{w} - \boldsymbol{w}_i \rangle \right] , \tag{6}$$

[1]http://svmlight.joachims.org/svm_struct.html

where $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t$ are the points at which the risk $R(\boldsymbol{w})$ is sampled and $R'(\boldsymbol{w}_i) \in \mathbb{R}^n$, $i = 1, \ldots, t$, denote 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 [Cheney and Goldstain, 1959] is a simple iterative procedure which exploits the CP model to solve the problem (5). Starting from an initial solution $\boldsymbol{w}_1 \in \mathbb{R}^n$, the CP algorithm computes the new iterates by solving the *reduced problem*

$$\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w} \in \mathbb{R}^n} R_t(\boldsymbol{w}) \, . \tag{7}$$

It is well known that the iterates generated by the CP algorithm show a strong zig-zag behavior, especially at the early iterations when the CP model is inaccurate, and that the CP algorithm converges very slowly.

The bundle methods [Lemaréchal, 1978] refine the CP algorithm by adding a quadratic prox-term to the reduced problem, i.e. the next iterate becomes

$$\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w} \in \mathbb{R}^n} \left[R_t(\boldsymbol{w}) + \alpha_i \|\boldsymbol{w} - \boldsymbol{w}_i^+\|^2 \right] , \tag{8}$$

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 iterates, thereby removing the detrimental zig-zag behavior of the CP algorithm. The bundle method is controlled by two rules. The first rule defines the minimal decrease threshold γ_t and, the second rule sets the prox-term penalty α_t. The two rules have a significant impact on the convergence of the bundle method [Lemaréchal et al., 1995]. The bundle methods are known to converge significantly faster than the CP algorithm.

[Teo et al., 2010] adopted the original bundle methods for the specific problem (1) whose objective already contains a quadratic term. In particular, they propose to replace the problem (1) by the following *reduced problem*

$$\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w} \in \mathbb{R}^n} F_t(\boldsymbol{w}) := \left[\frac{\lambda}{2} \|\boldsymbol{w}\|^2 + R_t(\boldsymbol{w}) \right] . \tag{9}$$

The reduced problem objective $F_t(\boldsymbol{w})$ is obtained from the primal problem objective $F(\boldsymbol{w})$ by replacing the risk $R(\boldsymbol{w})$ with its CP model $R_t(\boldsymbol{w})$ while the quadratic regularization term is unchanged. Hence, the regularization term serves as a natural prox-center. This is an elegant solution because it avoids designing rules for the updates of the prox-center penalty and the sufficient decrease threshold which are needed in the standard bundle methods. The method is termed the Bundle Methods for Risk Minimization (BMRM).

Starting from an initial guess $\boldsymbol{w}_1 \in \mathbb{R}^n$, the BMRM iteratively solves the reduced problem (9) and uses the new iterate $\boldsymbol{w}_{t+1}$ to update the CP model (6) which becomes progressively more accurate. This process is repeated until a gap between the primal and the reduced objective gets below a prescribed $\varepsilon > 0$, i.e., until $F(\boldsymbol{w}_{t+1}) - F_t(\boldsymbol{w}_{t+1}) \leq$

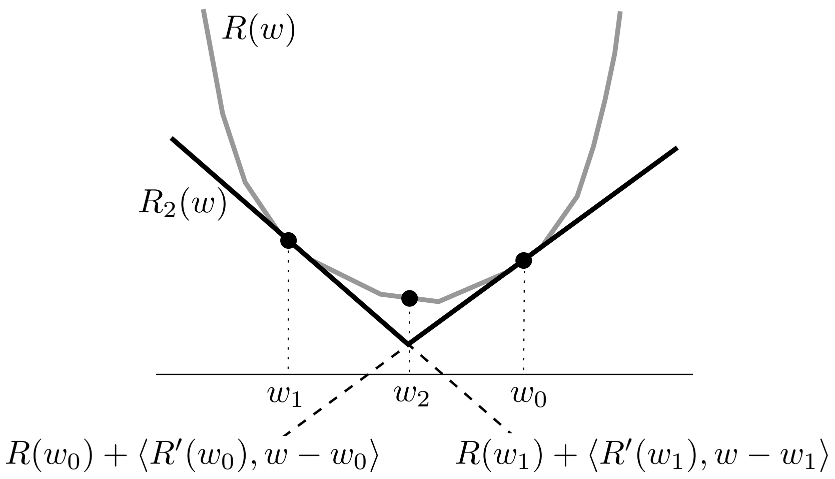

$R(w)$
$R_2(w)$
w_1
w_2
w_0
$R(w_0) + \langle R'(w_0), w - w_0 \rangle$
$R(w_1) + \langle R'(w_1), w - w_1 \rangle$

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

$\varepsilon|F(\boldsymbol{w}_{t+1})|$ holds, which implies that new iterate $\boldsymbol{w}_{t+1}$ is ε-optimal solution of (1) satisfying $F(\boldsymbol{w}_{t+1})(1-\varepsilon) \leq F(\boldsymbol{w}^*)$. It can be proved [Teo et al., 2010], that for an arbitrary $\varepsilon > 0$ the ε-optimal solution is obtained after at most $\mathcal{O}(\log_2 \lambda + \frac{C}{\lambda \epsilon})$ iterations, where C is a problem dependent constant which does not depend on λ and ε.

The simplicity of the BMRM algorithm is compensated by a reduced efficiency compared to the original bundle methods. The prox-term penalty is in the BMRM replaced by a fixed regularization parameter λ. Hence, for low values of λ the influence of the quadratic term is weak and the BMRM becomes closer to the CP algorithm, i.e. it exhibits a zig-zag behavior and very slow convergence. The detrimental effect of a low λ is also seen from the upper bound on the maximal number of iterations $\mathcal{O}(\log_2 \lambda + \frac{C}{\lambda \epsilon})$.

Efficiency of the BMRM algorithm thus depends on the overall number of iterations and the per iteration complexity. The per iteration time is typically dominated by computation of the risk value $R(\boldsymbol{w}_t)$ and its sub-gradient $R'(\boldsymbol{w}_t)$ while solving the reduced problem is cheap. The number of iterations the BMRM needs to converges strongly depends on the regularization constant λ. The quadratic regularizer $\frac{\lambda}{2}\|\boldsymbol{w}\|^2$ only helps to avoid over-fitting but it also speeds up the optimization. The quadratic term effectively constraints the space of the parameters (this can be easily shown formally). For higher values of λ the effect of the regularizer is stronger and, consequently, the BMRM needs less iterations to converge. On the other hand, for lower λ the iterates generated by the BMRM exhibit a strong zig-zag behavior resulting in unacceptably high number of iterations.

The BMRM algorithm (see Algorithm 1) typically requires few hundred iterations to converge. The evaluation of the risk $R(\boldsymbol{w})$ and its sub-gradient $R'(\boldsymbol{w})$ required in step 4 is very often the most expensive part of the algorithm.

Algorithm 1 BMRM

```
Require: $\epsilon$, a function pointer $R(\boldsymbol{w})$, a function pointer $R'(\boldsymbol{w})$
1: Initialization: $\boldsymbol{w} \leftarrow \boldsymbol{0}, t \leftarrow 0$
2: repeat
3:     $t \leftarrow t + 1$
4:     Compute $R(\boldsymbol{w}_t), R'(\boldsymbol{w}_t)$
5:     Update $R_t(\boldsymbol{w}_t)$
6:     Solve $\boldsymbol{w}_t \leftarrow \arg\min F_t(\boldsymbol{w})$
7:     $\epsilon_t \leftarrow F(\boldsymbol{w}_t) - F_t(\boldsymbol{w}_t)$
8: until $\epsilon_t \leq \epsilon$
```

3 Improved BMRM

In this section, we propose three speed improvements of the original BMRM which significantly improve its convergence. First, we propose to use the prox-term in the definition of the reduced problem. Its strength is adaptively adjusted in order to avoid the zig-zag behavior of the BMRM. Second, we propose to start the optimization from an approximate solution found by an on-line algorithm. As the third improvement, we propose to approximate the risk by the multiple cutting plane model instead of the single cutting plane model used by BMRM. These improvements are detailed in the following sections.

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 bundle methods. The difference when compared to the classical bundle methods is that we do not approximate the quadratic regularizer and we propose new rules for adjusting the prox-term penalty and the minimal improvement threshold.

The main motivation is to prevent overly big changes of the solution vector by requiring that the euclidean distance between two consecutive iterates $\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|$ is not larger than some reasonably chosen constant $K > 0$. To this end, we endow the objective function of the reduced problem by a prox-term, i.e. the modified objective of the reduced problem becomes

$$F_t(\boldsymbol{w}, \alpha) := \left[\frac{\lambda}{2} \|\boldsymbol{w}\|^2 + R_t(\boldsymbol{w}) + \alpha \|\boldsymbol{w} - \boldsymbol{w}_t\|^2 \right] , \tag{10}$$

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 (10) with the value of α set adaptively. The optimization schema is described in Algorithm 2.

In each iteration, the Prox-BMRM first tries to compute new iterate by minimizing the reduced objective with the prox-center penalty α used in the previous step (line 3). If the new iterate sufficiently improves the primal objective, i.e. the primal objective

Algorithm 2 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:
                $\boldsymbol{w}_{t+1} = \boldsymbol{w}_{t+1}^{\alpha_t}$
                $\alpha_{t+1} = \alpha_t$
                $\gamma_{t+1} = \gamma_t$
 6:     else
 7:         Find the minimal $\hat{\alpha} \in \{0, 1, 2, 4, \dots\}$ 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
                $\gamma_{t+1} = \frac{F(\boldsymbol{w}_{t+1}^{\hat{\alpha}})}{T} - \frac{F_t(\boldsymbol{w}_{t+1}^{0})}{T(1-\varepsilon)}$     (11)
 9:     end if
10: until $F(\boldsymbol{w}_{t+1}) - F_t(\boldsymbol{w}_{t+1}^{0}) \leq \varepsilon \cdot |F(\boldsymbol{w}_{t+1})|$
```

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 (line 5). 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} according to formula (11). 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 it 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 worm 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 as it requires solving the reduced problem anyway. With these tricks implemented, we observed that the reduced problem is solved on average 2-3 times instead of 1 times which constitutes a neglectable increase of computation time. This increase is amply compensated by the reduced number of 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 iterates and T which is the maximal number of iterations without readjusting the prox-center penalty α. An efficient and simple 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 2 reduces the solution of (1) to a sequence of problems (10). The problem (10) is equivalent to the following quadratic program (QP):

$$\begin{aligned} \boldsymbol{w}_{t+1} \;&=\; \arg\min_{\boldsymbol{w}\in\mathbb{R}^n} \left[\frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \alpha_t\|\boldsymbol{w} - \boldsymbol{w}_t\|^2 + \xi\right] \\ \text{s.t.} \quad \xi \;&\geq\; \boldsymbol{a}_i^\top \boldsymbol{w} + b_i,\ i = 0,\dots,t-1 \end{aligned} \tag{12}$$

where $\boldsymbol{a}_i$ is a sub-gradient $R'(\boldsymbol{w})$ and $b_i = R(\boldsymbol{w}_i) - \langle \boldsymbol{w}_i, R'(\boldsymbol{w}_i)\rangle$. In practice, the number of cutting planes t required by Algorithm 2 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. Let $\mathbf{A} = [\boldsymbol{a}_0, \dots, \boldsymbol{a}_{t-1}] \in \mathbb{R}^{n\times t}$ be a matrix, whose columns are sub-gradients and let $\boldsymbol{b} = [b_0, \dots, b_{t-1}] \in \mathbb{R}^t$ be a column vector. Then using the Wolfe duality, one can derive the Lagrange dual of (12) which reads as follows

$$\begin{aligned} \boldsymbol{\beta_t} \;&\in\; \arg\max_{\boldsymbol{\beta}\in\mathbb{R}^t} \left[-\frac{1}{2(\lambda + 2\alpha_t)}\boldsymbol{\beta}^\top \mathbf{A}^\top \mathbf{A}\boldsymbol{\beta} + \boldsymbol{\beta}^\top\left(\boldsymbol{b} + \frac{2\alpha_t \mathbf{A}^\top \boldsymbol{w}_t}{\lambda + 2\alpha_t}\right) + \frac{\alpha_t \lambda \boldsymbol{w}_t^\top \boldsymbol{w}_t}{\lambda + 2\alpha_t}\right] \\ \text{s.t.} \quad &\|\boldsymbol{\beta}\|_1 = 1,\ \boldsymbol{\beta} \geq 0\,. \end{aligned} \tag{13}$$

The primal solution can be obtained analytically by $\boldsymbol{w}_{t+1} = \frac{2\alpha_t \boldsymbol{w}_t - \mathbf{A}\boldsymbol{\beta}}{\lambda + 2\alpha_t}$. Note that since the last term in (13) does not depend on $\boldsymbol{\beta}$, we are solving the quadratic program of exactly the same form as in standard BMRM [Teo et al., 2010].

3.2 Initialization by on-line methods and warm start

The on-line algorithms show fast convergence in the early optimization stages but then it takes long time before they get to a close vicinity of the optimal solution. It is a straightforward idea to refine the imprecise solution obtained by an on-line algorithm by a precise method or to reuse solution that was found previously, e.g. in validation stage. To this end we use the following two strategies.

The first strategy is to use on-line algorithm. Particularly, we use a variant of the Stochastic Gradient Descent algorithm [Bordes et al., 2009]. We run 10 passes of the SGD algorithm and keep track of the minimal value of the primal objective $F(\boldsymbol{w})$. Finally, we take the SGD solution with the minimal objective value as the initial point of the Prox-BMRM algorithm.

The second strategy is to reuse previously obtained solutions. For example during validation stage, one typically needs to train the classifier with several values of regularization constant $\lambda \in \Lambda$, where $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$. 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 $\boldsymbol{w}^{\lambda_i}$ obtained for λ_i as an initial solution for computing $\boldsymbol{w}^{\lambda_{i+1}}$ with λ_{i+1}. Experimental evaluation shows that this strategy leads to even bigger speedups (up to approximately 10 times faster convergence), especially for lower values of regularization constant λ.

3.3 P-BMRM

As the third improvement, we propose to uses $P > 1$ cutting plane models to approximate the risk $R(\boldsymbol{w})$ instead of the single cutting plane model as in the original. We call the proposed method P-BMRM.

We assume the risk to be decomposed into $P > 1$ terms

$$R(\boldsymbol{w}) = \sum_{p=1}^{P} R(\boldsymbol{w}, p)$$

For example, one can evenly split the m training examples into P groups and then we set

$$R(\boldsymbol{w}, p) = \sum_{i \in I_p} r_i(\boldsymbol{w}) \ ,$$

where $I_1, \dots, I_P$ are mutually disjoint sets and $I_1 \cap \cdots \cap I_P = \{1, \dots, m\}$.

We propose to approximate each of the P partial risks $R(\boldsymbol{w}, p)$ by its own cutting plane model

$$R_t(\boldsymbol{w}, p) = \max_{i=1,\dots,t} [R(\boldsymbol{w}_i, p) + \langle R'(\boldsymbol{w}_i, p), \boldsymbol{w} - \boldsymbol{w}_i \rangle]$$

where $R'(\boldsymbol{w}_i, p)$ denotes a sub-gradient of $R(\boldsymbol{w}_i, p)$ at $\boldsymbol{w}_i$. The cutting plane model of the risk $R(\boldsymbol{w})$ is then

$$R_t(\boldsymbol{w}) = \sum_{p=1}^{P} R_t(\boldsymbol{w}, p). \tag{14}$$

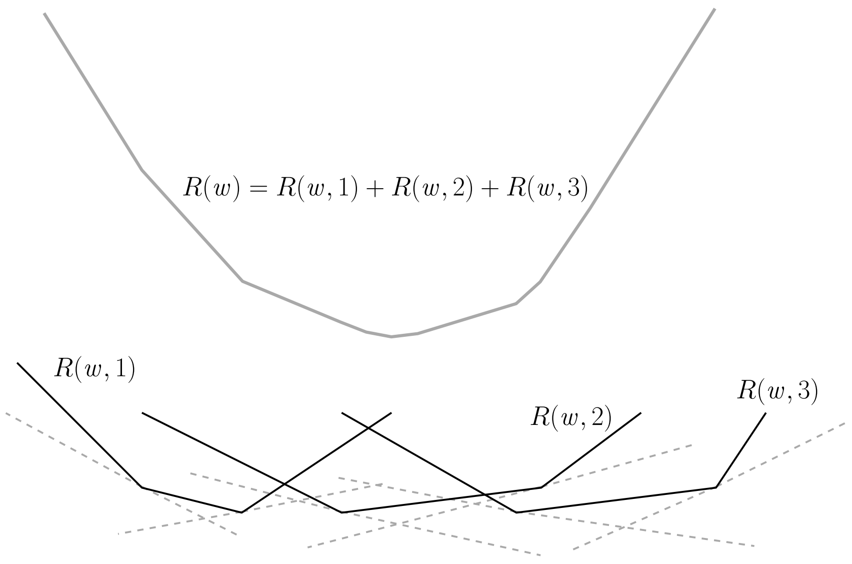


Figure 2: The figure illustrates how is the risk $R(\boldsymbol{w})$ (upper figure) decomposed into a sum of functions $R(\boldsymbol{w}, p)$, $p = 1, \ldots, P$, (lower figure) each of which is approximated by its own cutting planes (dashed lines).

We denote (14) as P-cutting plane model. Note that for $P = 1$, the P-cutting plane model (14) reduces to the original model (6). It is immediately seen that the P-cutting plane model (14) preserves the crucial properties of the original model, i.e., $R_t(\boldsymbol{w})$ is a lower bound of $R(\boldsymbol{w})$ which is tight at the points $\boldsymbol{w}_i$, $i = 1, \ldots, t$. Figure 2 illustrates the idea.

The P-BMRM is obtained just by substituting the P-cutting plane model for the original cutting plane model (6) in the definition of the reduced problem (9).

The updating of the P-cutting plane model has the same computational complexity as in the original BMRM. The higher accuracy is compensated by increase in memory requirements due to storing P times more cutting planes. The associated reduced problem can be expressed as the equivalent QP. In its dual form the QP has P constraints instead of a single one as in the original BMRM. However, the constraints are decoupled which allows using of sequential minimal solvers. The QP task is very similar to the one used in standard BMRM.

4 Experiments

In this section we experimentally evaluate the proposed improvements of the BMRM. We use three different instances of the SO-SVM learning as benchmarks. In particular, we consider the problem of learning the OCR classifier, the facial landmark detector and segmentation of car number plate images. We used the standard MNIST database for the OCR problem while the data for the last two problems originate from real life applications. We first briefly describe the benchmarks and then we present the evaluation.

Table 1: The table shows validation risks for all benchmarks as function of λ. The test risk is computed for the classifier with the minimal validation risk. For the OCR the risk is the classification error. For the face landmark detection the risk is the mean deviation of the estimated landmark positions where 100 corresponds to the distance between the center of eyes and the mouth. For the license plates the risk corresponds to the percentage of incorrectly segmented columns of the input image.

Benchmark 1: OCR — MNIST

	$\lambda = 10^3$	$\lambda = 10^2$	$\lambda = 10^1$	$\lambda = 10^0$	tst
val	0.0864	0.0740	0.0708	0.0720	0.0704

Benchmark 2: Facial landmark detection

	$\lambda = 10^4$	$\lambda = 10^3$	$\lambda = 10^2$	$\lambda = 10^1$	tst
val	11.03	6.36	5.46	5.80	5.46

Benchmark 3: License plate segmentation

	$\lambda = 10^5$	$\lambda = 10^4$	$\lambda = 10^3$	$\lambda = 10^2$	tst
val	22.22	13.16	7.02	4.61	4.21

4.1 OCR

As the first benchmark we consider the optical character recognition (OCR) problem. We use the MNIST database[2] 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. $y \in \mathcal{Y} = \{0, \dots, 9\}$. 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 (2) 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 $\boldsymbol{w}_y$, $y \in \mathcal{Y}$, represented themselves as columns vectors. 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 (2) becomes an instance of a linear multi-class SVM classifier.

The parameter vector has $n = 7,840$ components. We train on all $m = 60,000$ training examples and we use the test part of the database for validation and testing ($5,000$ examples each).

4.2 Facial landmark detection

As the second benchmark we consider learning of a face landmark detector. We follow the approach of [Uřičář et al., 2012] where the landmark detection is posed as an instance of the SO-SVM classifier (2). The classifier input $x \in \mathcal{X}$ is an image 40×40 pixels large which contains a face. The classifier outputs $y = (y^1, \dots, y^L) \in \mathcal{Y} = \mathcal{N}^{2\times L}$ which is a set of 2D coordinates of L landmarks like corners of the eyes, corners of the mouth, tip of the nose and the center of the face. The scoring function $\langle \boldsymbol{w}, \boldsymbol{\Psi}(x, y)\rangle$ of the classifier (2) is composed of

[2]`http://yann.lecun.com/exdb/mnist/`

the appearance model and the deformation cost. The appearance model evaluates a match between the input image x and the landmark templates put at positions y. The deformation cost evaluates likeliness of the particular landmark configuration y and this cost decomposes to a set of pair wise terms defined over edges of an acyclic graph. The map $\boldsymbol{\Psi}(x, y)$ is as a column-wise concatenation of local feature descriptors of individual landmarks and parameters of the deformation cost. We use a variant of Local Binary Patterns as the feature descriptor. Evaluation of the classifier (2) leads to solving an instance of the dynamic programing. The loss function measures the mean deviation between the ground truth landmark positions y and their estimate 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 from a set of $m = 7,000$ images with manually annotated landmark positions. In our experiment the number of landmarks was $L = 8$ and the number of parameters $\boldsymbol{w} \in \mathbb{R}^n$ to learn was $n = 232,476$.

4.3 Number plate segmentation

As the third benchmark we consider segmentation of car number plate images. 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

$$\left.\begin{array}{l} k(s_1) = 1\,, W = k(s_L) + \omega(s_L) - 1\,, \\ k(s_i) = k(s_{i-1}) + \omega(s_{i-1})\,, \forall i > 1\,, \end{array}\right\} \tag{15}$$

where $\omega\colon A \to \mathcal{N}$ are widths of the characters. The constraints (15) 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 (2) computes the correlation between the image x and the character templates placed one by one according to the segmentation $y \in Y$, i.e. $\langle \boldsymbol{\Psi}(x, y), \boldsymbol{w} \rangle$ equals to

$$\sum_{i=1}^{L(y)} \sum_{j=1}^{\omega(a(s_i))} \langle \mathrm{col}(x, j + k(s_i) - 1), \mathrm{col}(\boldsymbol{w}_{a(s_i)}, j) \rangle\,,$$

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 (2), as well as evaluation of $r_i(\boldsymbol{w})$ and its sub-gradient $\boldsymbol{r}_i(\boldsymbol{w})$, leads to an instance of the dynamic programming.

We used $m = 6,788$ annotated images for training and $1,692$ images for computing validation and test error. The parameter vector $\boldsymbol{w}$ had $n = 4,059$ components.

Table 2: 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. The speed up is computed by dividing the BMRM wall clock time by the time of the respective algorithm. The different strategies used for obtaining the initial solution are denoted in the upper left corner of each table. Initial solution SGD stands for obtaining the solution by pre-training the classifier with 10 passes of SGD, whereas REUSE stands for reusing the solution from the previous regularization constant λ (we have used $\lambda_0 = 10\lambda_1$ for this purpose to fairly compare with SGD strategy). Finally, different implementations have been used for benchmarks. Benchmark 1 was run using the SHOGUN implementation and its python modular interface, while Bnechmark 2 and 3 were run using the MATLAB implementation.

Benchmark 1: OCR - MNIST (SHOGUN)

Initial solution	$\lambda_1 = 1000$			$\lambda_2 = 100$			$\lambda_3 = 10$			$\lambda_4 = 1$		
REUSE	iter	time	spd	iter	time	spd	iter	time	spd	iter	time	spd
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)

Initial solution	$\lambda_1 = 10000$			$\lambda_2 = 1000$			$\lambda_3 = 100$			$\lambda_4 = 10$		
SGD	iter	time	spd	iter	time	spd	iter	time	spdup	iter	time	spd
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)

Initial solution	$\lambda_1 = 10^5$			$\lambda_2 = 10^4$			$\lambda_3 = 10^3$			$\lambda_4 = 10^2$		
SGD	iter	time	spd	iter	time	spd	iter	time	spd	iter	time	spd
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

4.4 Results

We compare the original BMRM with the proposed improved variants. We use Prox-BMRM to denote Algorithm 2 using the single cutting plane model to approximate the risk. The P-BMRM denotes the standard BMRM using P cutting plane models for risk approximation. Finally, the Prox-P-BMRM denotes the algorithm combining both improvements together.

In all experiments, we set the precision parameter to $\varepsilon = 0.01$. In experiments we have used both pure Matlab and `C++` implementation of tested methods. To this end `C++` implementation (from Shogun optimization toolbox [Sonnenburg et al., 2010]) was used only on the OCR-MNIST problem. All tested method use the same code for computing the risk value and its sub-gradient. The reduced problems are optimized by the same QP solver (LibQP [Franc and Hlaváč, 2006]).

We have experimented with two different strategies providing initial solution to the Prox-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 on-line learning algorithm SGD.

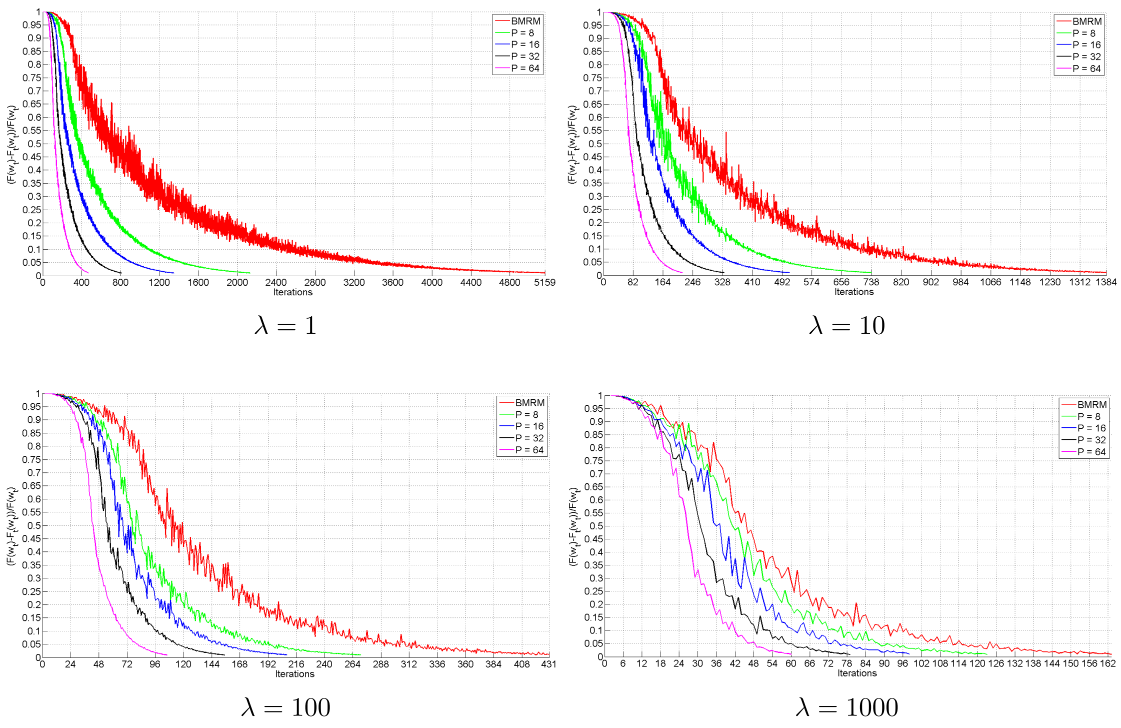
We learned the SO-SVM classifier for a range of regularization parameters λ on the training set and for each classifier we computed the validation risk on the validation set. Then the test examples were used to compute the test risk for the classifier with the minimal validation risk. In Table 1 we report the obtained validation and test risks for all benchmarks. Due the low precision parameter $\varepsilon = 0.01$ the risk values are the same for all algorithms. The risk values differed on fourth place after the decimal point.

All tested algorithms provide the same precise classifier but they require different time to converge. We measured the time to convergence in terms of i) the number of iterations and ii) the wall clock time. The obtained results are summarized in Table 2. We see that the Prox-BMRM significantly decreases the number of iterations as well as 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 improvement is best seen on the license plate benchmark where the parameters have relatively low dimension ($n = 4,059$). Using more CP models is less beneficial on the face landmark problem where the data are high dimensional ($n = 232,476$) and sparse. The maximal speedup 9.7 was obtained on the OCR-MNIST problem for the lowest λ and Prox-P=16-BMRM algorithm.

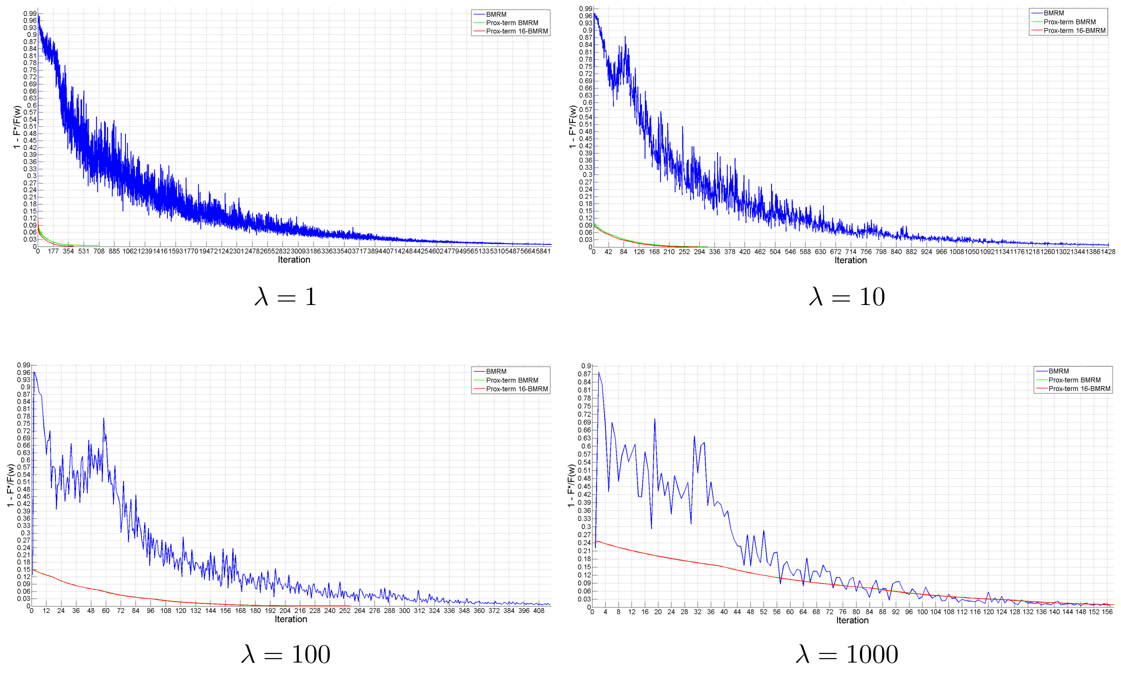
To show the effect of the individual improvements we plot convergence curves for the OCR-MNIST benchmark. Figure 3(a) shows the convergence curves for the original BMRM and the P-BMRM with increasing value of P. It is seen that the number of iterations steadily decreases with growing number of the CP models. The optimal P depends on the memory available and the overhead introduced by solving more complex reduced problem. We found value $P = 16$ to produce the best results in our experiments. Figure 3(b) compares convergence curves for the BMRM and the Prox-BMRM. As expected the convergence curve of the Prox-BMRM behaves much more reasonably compared to the BMRM whose curves strongly fluctuates.

5 Conclusions

In this paper we have analyzed a source of inefficiency of the BMRM and propose three improvements which significantly speedup its convergence. The first improvement brings the BMRM back to the classical bundle methods from which it has been derived. In particular, we prose to use a 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 found by an imprecise on-line algorithm like the SGD, or the solution found with previous value of regularization constant λ can be readily used. Finally, we propose to use multiple cutting plane model which allows a finer approximation the of risk term. We evaluate the proposed improvements on one standard benchmark and two real-life applications of the SO-SVM classifiers. The experiments show that the BMRM using the proposed improvements converges significantly faster achieving speedup up to a factor of 9.7 compared to the original BMRM.



(a)



(b)

Figure 3: Convergence curves for learning the SO-SVM classifier on the OCR-MNIST benchmark with different values of λ. Figure (a) shows curves for the original BMRM and the P-BMRM with different values of P. Figure (b) shows curves for the original BMRM and the Prox-BMRM algorithm. The x-axis is the iteration counter while the y-axis shows the relative distance of the current solution to the optimal value.

References

[Bordes et al., 2009] Bordes, A., Bottou, L., and Gallinari, P. (2009). SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent. *Journal of Machine Learning Research*, 10:1737–1754.

[Cheney and Goldstain, 1959] Cheney, E. and Goldstain, A. (1959). Newton's method for convex programming and tchebytcheff approximation. *Numerische Mathematick*, 1:253–268.

[Franc and Hlaváč, 2006] Franc, V. and Hlaváč, V. (2006). A novel algorithm for learning support vector machines with structured output spaces. Research Report K333–22/06, CTU–CMP–2006–04, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University, Prague, Czech Republic.

[Joachims et al., 2009] Joachims, T., Finley, T., and Yu, C.-N. (2009). Cutting-Plane Training of Structural SVMs. *Machine Learning*, 77(1):27–59.

[Lemaréchal, 1978] Lemaréchal, C. (1978). Nonsmooth optimization and descend methods. Technical report, IIASA, Laxenburg, Austria.

[Lemaréchal et al., 1995] Lemaréchal, C., Nemirovskii, A., and Nesterov, Y. (1995). New variants of bundle methods. *Mathematical Programming*, 69:111–147.

[Shwartz et al., 2007] Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 807 – 814. ACM Press.

[Sonnenburg et al., 2010] Sonnenburg, S., Rätsch, G., Henschel, S., Widmer, C., Behr, J., Zien, A., Bona, F. d., Binder, A., Gehl, C., and Franc, V. (2010). The shogun machine learning toolbox. *J. Mach. Learn. Res.*, 99:1799–1802.

[Teo et al., 2010] Teo, C., Vishwanathan, S., Smola, A., and Quoc, V. (2010). Bundle Methods for Regularized Risk Minimization. *Journal of Machine Learning Research*, 11:311–365.

[Tsochantaridis et al., 2005] Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research*, 6:1453–1484.

[Uřičář et al., 2012] Uřičář, M., Franc, V., and Hlaváč, V. (2012). Detector of Facial Landmarks Learned by the Structured Output SVM. In Csurka, G. and Braz, J., editors, *VISAPP '12: Proceedings of the 7th International Conference on Computer Vision Theory and Applications*, volume 1, pages 547–556, Portugal. SciTePress — Science and Technology Publications.