Journal of Machine Learning Research 13 (2012) 3623-3680 Submitted 11/11; Revised 9/12; Published 12/12

Smoothing Multivariate Performance Measures

Xinhua Zhang[*] XINHUA2@CS.UALBERTA.CA
Department of Computing Science
University of Alberta
Alberta Innovates Center for Machine Learning
Edmonton, Alberta T6G 2E8, Canada

Ankan Saha ANKANS@CS.UCHICAGO.EDU
Department of Computer Science
University of Chicago
Chicago, IL 60637, USA

S.V. N. Vishwanathan VISHY@STAT.PURDUE.EDU
Departments of Statistics and Computer Science
Purdue University
West Lafayette, IN 47907-2066, USA

Editor: Sathiya Keerthi

Abstract

Optimizing multivariate performance measure is an important task in Machine Learning. Joachims (2005) introduced a Support Vector Method whose underlying optimization problem is commonly solved by cutting plane methods (CPMs) such as SVM-Perf and BMRM. It can be shown that CPMs converge to an ε accurate solution in $O\left(\frac{1}{\lambda\varepsilon}\right)$ iterations, where λ is the trade-off parameter between the regularizer and the loss function. Motivated by the impressive convergence rate of CPM on a number of practical problems, it was conjectured that these rates can be further improved. We disprove this conjecture in this paper by constructing counter examples. However, surprisingly, we further discover that these problems are not inherently hard, and we develop a novel smoothing strategy, which in conjunction with Nesterov's accelerated gradient method, can find an ε accurate solution in $O^*\left(\min\left\{\frac{1}{\varepsilon}, \frac{1}{\sqrt{\lambda\varepsilon}}\right\}\right)$ iterations. Computationally, our smoothing technique is also particularly advantageous for optimizing multivariate performance scores such as precision/recall break-even point and ROCArea; the cost per iteration remains the same as that of CPMs. Empirical evaluation on some of the largest publicly available data sets shows that our method converges significantly faster than CPMs without sacrificing generalization ability.

Keywords: non-smooth optimization, max-margin methods, multivariate performance measures, Support Vector Machines, smoothing

1. Introduction

Recently there has been an explosion of interest in applying machine learning techniques to a number of application domains, much of which has been fueled by the phenomenal success of binary Support Vector Machines (SVMs). At the heart of SVMs is the following regularized risk mini-

∗. Xinhua Zhang is now working at the Machine Learning Group of NICTA, Canberra, Australia.

mization problem:

$$\min_{\mathbf{w}} J(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + R_{\text{emp}}(\mathbf{w}). \tag{1}$$

Here R_{emp} is the so-called empirical risk (see below), $\frac{1}{2}\|\mathbf{w}\|^2$ is the regularizer, $\lambda > 0$ is a scalar which trades off the importance of the empirical risk and the regularizer. Given a training set with n examples $\mathcal{X} := \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \{\pm 1\}$, the empirical risk R_{emp} minimized by SVMs is the average of the hinge loss:

$$R_{\text{emp}}(\mathbf{w}) := \frac{1}{n}\sum_{i=1}^n \max(0, 1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle), \tag{2}$$

where $\langle \cdot, \cdot \rangle$ denotes the Euclidean dot product.

Binary SVMs optimize classification accuracy, while many application areas such as Natural Language Processing (NLP) frequently use more involved multivariate performance measures such as precision/recall break-even point (PRBEP) and area under the Receiver Operating Characteristic curve (ROCArea). Joachims (2005) proposed an elegant SVM based formulation for *directly* optimizing these performance metrics. In his formulation, the empirical risk R_{emp} in (1) is replaced by

$$R_{\text{emp}}(\mathbf{w}) := \max_{\mathbf{z} \in \{-1,1\}^n} \left[\Delta(\mathbf{z}, \mathbf{y}) + \frac{1}{n}\sum_{i=1}^n \langle \mathbf{w}, \mathbf{x}_i \rangle (z_i - y_i) \right]. \tag{3}$$

Here, $\Delta(\mathbf{z}, \mathbf{y})$ is the multivariate discrepancy between the correct labels $\mathbf{y} := (y_1, \ldots, y_n)^\top$ and a candidate labeling $\mathbf{z} := (z_1, \ldots, z_n)^\top \in \{\pm 1\}^n$. In order to compute the multivariate discrepancy for PRBEP we need the false negative and false positive rates, which are defined as

$$b = \sum_{i \in \mathcal{P}} \delta(z_i = -1) \quad \text{and} \quad c = \sum_{j \in \mathcal{N}} \delta(z_j = 1), \text{ respectively.}$$

Here $\delta(x) = 1$ if x is true and 0 otherwise, while $\mathcal{P}$ and $\mathcal{N}$ denote the set of indices with positive ($y_i = +1$) and negative ($y_i = -1$) labels respectively. Furthermore, let $n_+ = |\mathcal{P}|$, $n_- = \left|\mathcal{N}\right|$. With this notation in place, $\Delta(\boldsymbol{z}, \boldsymbol{y})$ for PRBEP is defined as

$$\Delta(\boldsymbol{z}, \boldsymbol{y}) = \begin{cases} b/n_+ & \text{if } b = c \\ -\infty & \text{otherwise} \end{cases}. \tag{4}$$

ROCArea, on the other hand, measures how many pairs of examples are mis-ordered. Denote $m = n_+ n_-$. Joachims (2005) proposed using the following empirical risk, R_{emp}, to directly optimize the ROCArea:[1]

$$R_{\text{emp}}(\mathbf{w}) = \frac{1}{m} \max_{\mathbf{z} \in \{0,1\}^m} \left[\sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} z_{ij} \left[1 - \mathbf{w}^\top (\mathbf{x}_i - \mathbf{x}_j) \right] \right]. \tag{5}$$

We will call the regularized risk minimization problem with the PRBEP loss (3) as the PRBEP-problem and with the ROCArea loss (5) as the ROCArea-problem respectively. As is obvious,

1. The original formulation uses $1 - 2\mathbf{w}^\top(\mathbf{x}_i - \mathbf{x}_j)$, and we rescaled it to simplify our presentation in the sequel.

both these problems entail minimizing a non-smooth objective function which is *not* additive over the data points. Fortunately, cutting plane methods (CPMs) such as SVM-Perf (Joachims, 2006) and BMRM (Teo et al., 2010) can be used for this task. At each iteration these algorithms only require a sub-gradient of R_{emp}, which can be efficiently computed by a *separation* algorithm with $O(n \log n)$ effort for both (3) and (5) (Joachims, 2005). In this paper, we will work with BMRM as our prototypical CPM, which, as Teo et al. (2010) point out, includes SVM-Perf as a special case.

CPMs are popular in Machine Learning because they come with strong convergence guarantees. The tightest upper bound on the convergence speed of BMRM, which we are aware of, is due to Teo et al. (2010). Theorem 5 of Teo et al. (2010) asserts that BMRM can find an ε-accurate solution of the regularized risk minimization problem (1) with non-smooth empirical risk functions such as (3) and (5) after computing $O(\frac{1}{\lambda\varepsilon})$ subgradients. This upper bound is built upon the following standard assumption:

Assumption 1 (A1) *The subgradient of R_{emp} is bounded, that is, at any point $\mathbf{w}$, there exists a subgradient $\boldsymbol{g} \in \partial R_{\text{emp}}(\mathbf{w})$ such that $\|\boldsymbol{g}\| \leq G < \infty$.*

In practice the observed rate of convergence of BMRM is significantly faster than predicted by theory. Therefore, it was conjectured that perhaps a more refined analysis might yield a tighter upper bound (Teo et al., 2010). Our first contribution in this paper is to disprove this conjecture under assumption **A1**. We carefully construct PRBEP and ROCArea problems on which BMRM requires at least $\Omega(1/\varepsilon)$ iterations to converge for a fixed λ. It is worthwhile emphasizing here that due to the specialized form of the objective function in (1), lower bounds for *general* convex function classes such as those studied by Nesterov (2003) and Nemirovski and Yudin (1983) do not apply. In addition, our examples stick to binary classification and are substantially different from the one in Joachims et al. (2009) which requires an increasing number of classes.

This result leads to the following natural question: do the lower bounds hold because the PRBEP and ROCArea problems are fundamentally hard, or is it an inherent limitation of CPM? In other words, does there exist a solver which solves the PRBEP and ROCArea problems by invoking the separation algorithm for fewer than $O(\frac{1}{\lambda\varepsilon})$ times?[2] We provide partial answers. To understand our results one needs to understand another standard assumption that is often made when proving convergence rates:

Assumption 2 (A2) *Each $\mathbf{x}_i$ lies inside an L_2 (Euclidean) ball of radius R, that is, $\|\mathbf{x}_i\| \leq R$.*

Clearly assumption **A2** is more restrictive than **A1**, because $\|\boldsymbol{x}_i\| \leq R$ implies that for all $\boldsymbol{w}$, any subgradient of R_{emp} has L_2 norm at most $2R$ in both (3) and (5). Our second contribution in this paper is to show that the $O(\frac{1}{\lambda\varepsilon})$ barrier can be broken under assumption **A2**, while making a similar claim just under assumption **A1** remains an open problem. In a nutshell, our algorithm approximates (1) by a smooth function, which in turn can be efficiently minimized by using either accelerated gradient descent (Nesterov, 1983, 2005, 2007) or a quasi-Newton method (Nocedal and Wright, 2006). This technique for non-smooth optimization was pioneered by Nesterov (2005). However, applying it to multivariate performance measures requires special care. We now describe some relevant details and point out technical difficulties that one encounters along the way.

2. We want the rates to be better in both λ *and* in ε.

1.1 Nesterov's Formulation

Let A be a linear transform and assume that we can find a smooth function $g^\star_\mu(A^\top \mathbf{w})$ with a Lipschitz continuous gradient such that $\left|R_{\text{emp}}(\mathbf{w}) - g^\star_\mu(A^\top \mathbf{w})\right| \leq \mu$ for all $\mathbf{w}$. It is easy to see that

$$J_\mu(\mathbf{w}) := \frac{\lambda}{2}\|\mathbf{w}\|^2 + g^\star_\mu(A^\top \mathbf{w}) \tag{6}$$

satisfies $\left|J_\mu(\mathbf{w}) - J(\mathbf{w})\right| \leq \mu$ for all $\mathbf{w}$. In particular, if we set $\mu = \varepsilon/2$ and find a $\mathbf{w}'$ such that $J_\mu(\mathbf{w}') \leq \min_{\mathbf{w}} J_\mu(\mathbf{w}) + \varepsilon/2$, then it follows that $J(\mathbf{w}') \leq \min_{\mathbf{w}} J(\mathbf{w}) + \varepsilon$. In other words, $\mathbf{w}'$ is an ε accurate solution for (1).

By applying Nesterov's accelerated gradient method to $J_\mu(\mathbf{w})$ one can find an ε-accurate solution of the original nonsmooth problem (1) after querying the gradient of $g^\star_\mu(A^\top \mathbf{w})$ at most

$$O^*\left(\sqrt{D}\,\|A\| \min\left\{\frac{1}{\varepsilon}, \frac{1}{\sqrt{\lambda\varepsilon}}\right\}\right) \tag{7}$$

number of times (Nesterov, 1983, 2005).[3] Here $\|A\|$ is the matrix norm of A, which is defined as $\|A\| = \max_{\boldsymbol{u},\boldsymbol{v}:\|\boldsymbol{u}\|=\|\boldsymbol{v}\|=1} \boldsymbol{u}^\top A \boldsymbol{v} = \sqrt{\lambda_{\max}(A^\top A)}$, with $\lambda_{\max}$ denoting the maximum eigenvalue of A, when the norm considered is the Euclidean norm. Furthermore, D is a geometric constant that depends solely on g_μ and is independent of ε, λ, or A. Compared with the $O(\frac{1}{\lambda\varepsilon})$ rates of CPMs, the $\frac{1}{\sqrt{\lambda\varepsilon}}$ part in (7) is already superior. Furthermore, many applications require $\lambda \ll \varepsilon$ and in this case the $\frac{1}{\varepsilon}$ part of the rate is even better. Note that CPMs rely on $\frac{\lambda}{2}\|\mathbf{w}\|^2$ to stabilize each update, and it has been empirically observed that they converge slowly when λ is small (see, e.g., Do et al., 2009).

Although the above scheme is conceptually simple, the smoothing of the objective function in (1) has to be performed very carefully in order to avoid dependence on n, the size of the training set. The main difficulties are two-fold. First, one needs to obtain a smooth approximation $g^\star_\mu(A^\top \mathbf{w})$ to $R_{\text{emp}}(\mathbf{w})$ such that $\sqrt{D}\,\|A\|$ is small (ideally a constant). Second, we need to show that computing the gradient of $g^\star_\mu(A^\top \mathbf{w})$ is no harder than computing a subgradient of $R_{\text{emp}}(\mathbf{w})$. In the sequel we will demonstrate how both the above difficulties can be overcome. Before describing our scheme in detail we would like to place our work in context by discussing some relevant related work.

1.2 Related Work

Training large SV models by using variants of stochastic gradient descent has recently become increasingly popular (Bottou, 2008; Shalev-Shwartz et al., 2007). While the dependence on ε and λ is still $\Omega(\frac{1}{\lambda\varepsilon})$ or worse (Agarwal et al., 2009), one gets bounds on the computational cost independent of n. However, stochastic gradient descent can only be applied when the empirical risk is *additively decomposable*, that is, it can be written as the average loss over individual data points like in (2). Since the non-linear multivariate scores such as the ones that we consider in this paper are not additively decomposable, this rules out the application of online algorithms to these problems.

Traditionally, batch optimizers such as the popular Sequential Minimal Optimization (SMO) worked in the dual (Platt, 1998). However, as List and Simon (2009) show, for the empirical risk (2) based on hinge loss, if the matrix which contains the dot products of all training instances

3. For completeness we reproduce relevant technical details from Nesterov (1983) and Nesterov (2005) in Appendix A.

(also known as the kernel matrix) is not strictly positive definite, then SMO requires $O(n/\varepsilon)$ iterations with each iteration costing $O(np)$ effort. However, when the kernel matrix is strictly positive definite, then one can obtain an $O(n^2 \log(1/\varepsilon))$ bound on the number of iterations, which has better dependence on ε, but is prohibitively expensive for large n. Even better dependence on ε can be achieved by using interior point methods (Ferris and Munson, 2002) which require only $O(\log(\log(1/\varepsilon))$ iterations, but the time complexity per iteration is $O(\min\{n^2 p, p^2 n\})$.

Recently, there has been significant research interest in optimizers which directly optimize (1) because there are some distinct advantages (Teo et al., 2010). Chapelle (2007) observed that to find a $\mathbf{w}$ which generalizes well, one only needs to solve the primal problem to very low accuracy (e.g., $\varepsilon \approx 0.001$). In fact, to the best of our knowledge, Chapelle (2007) introduced the idea of smoothing the objective function to the Machine Learning community. Specifically, he proposed to approximate the binary hinge loss by a smooth Huber's loss and used the Newton's method to solve this smoothed problem. This approach yielded the best overall performance in the Wild Competition Track of Sonnenburg et al. (2008) for training binary linear SVMs on large data sets. A similar smoothing approach is proposed by Zhou et al. (2010), but it is also only for hinge loss.

However, the smoothing proposed by Chapelle (2007) for the binary hinge loss is rather ad-hoc, and does not easily generalize to (3) and (5). Moreover, a function can be smoothed in many different ways and Chapelle (2007) did not explicitly relate the influence of smoothing on the rates of convergence of the solver. In contrast, we propose principled approaches to overcome these problems.

Of course, other smoothing techniques have also been explored in the literature. A popular approach is to replace the nonsmooth max term by a smooth log-sum-exp approximation (Boyd and Vandenberghe, 2004). In the case of binary classification this approximation is closely related to logistic regression (Bartlett et al., 2006; Zhang, 2004), and is equivalent to using an entropy regularizer in the dual. However, as we discuss in Section 3.1.2 this technique yields exorbitantly large $\sqrt{D}\,\|A\|$ in the bound (7) when applied to optimize multivariate performance measures.

1.3 Notation And Paper Outline

We assume a standard setup as in Nesterov (2005). Lower bold case letters (e.g., $\mathbf{w}$, $\boldsymbol{\mu}$) denote vectors, w_i denotes the i-th component of $\mathbf{w}$, $\mathbf{0}$ refers to the vector with all zero components, $\boldsymbol{e}_i$ is the i-th coordinate vector (all 0's except 1 at the i-th coordinate) and Δ_k refers to the k dimensional simplex. Unless specified otherwise, $\|\cdot\|$ refers to the Euclidean norm $\|\mathbf{w}\| := \sqrt{\langle \boldsymbol{w}, \boldsymbol{w} \rangle}$. We denote $\overline{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$, and $[t] := \{1, \ldots, t\}$. $g^\star$ stands for the Fenchel dual of a function g, and $\partial g(\boldsymbol{w})$ is the subdifferential of g at $\boldsymbol{w}$ (the set of all subgradients of g at $\boldsymbol{w}$).

In Section 2, we first establish the lower bounds on the rates of convergence of CPMs when used to optimize the regularized risk of PRBEP and ROCArea. To break this lower bound, our proposed strategy constructs a smoothed surrogate function $g_\mu^\star(A^\top \mathbf{w})$, and the details are given in Section 3. We will focus on efficiently computing the gradient of the smooth objective function in Section 4. Empirical evaluation is presented in Section 5, and the paper concludes with a discussion in Section 6.

1.4 Improvement On Earlier Versions Of This Paper

This paper is based on two prior publications of the same authors: Zhang et al. (2011a) at NIPS 2010 and Zhang et al. (2011b) at UAI 2011, together with a technical report (Zhang et al., 2010). Major improvements have been made on them, which include:

- New lower bounds for PRBEP and ROCArea losses (see Sections 2.2 and 2.3 respectively).
- Significantly simplified algorithms for computing the gradient of smoothed objectives for both PRBEP and ROCArea losses (See Sections 4.1 and 4.2 respectively).
- Empirical evaluation on some of the largest publicly available data sets including those from the Pascal Large Scale Learning Workshop in Section 5 (Sonnenburg et al., 2008)
- Open source code based on PETSc (Balay et al., 2011) and TAO (Benson et al., 2010) packages made available for download from Zhang et al. (2012).

2. Lower Bounds For Cutting Plane Methods

BMRM (Teo et al., 2010) is a state-of-the-art cutting plane method for optimizing multivariate performance scores which directly minimizes the non-smooth objective function $J(\boldsymbol{w})$. At every iteration, BMRM replaces R_{emp} by a piecewise linear lower bound R_k^{cp} and obtains the next iterate $\mathbf{w}_k$ by optimizing

$$\mathbf{w}_k := \operatorname*{argmin}_{\boldsymbol{w}} J_k(\mathbf{w}) := \frac{\lambda}{2}\|\mathbf{w}\|^2 + R_k^{\text{cp}}(\mathbf{w}), \quad \text{where } R_k^{\text{cp}}(\mathbf{w}) := \max_{i\in[k]} \langle \mathbf{w}, \mathbf{a}_i \rangle + b_i. \tag{8}$$

Here $\mathbf{a}_i \in \partial R_{\text{emp}}(\mathbf{w}_{i-1})$ denotes an *arbitrary* subgradient of R_{emp} at $\mathbf{w}_{i-1}$ and $b_i = R_{\text{emp}}(\mathbf{w}_{i-1}) - \langle \mathbf{w}_{i-1}, \mathbf{a}_i \rangle$. The piecewise linear lower bound is successively tightened until the gap

$$\varepsilon_k := \min_{0\le t\le k} J(\mathbf{w}_t) - J_k(\mathbf{w}_k) \tag{9}$$

falls below a predefined tolerance ε.

Since J_k in (8) is a convex objective function, the optimal $\boldsymbol{w}_k$ can be obtained by solving its dual problem, which is a quadratic programming problem with simplex constraints. We refer the reader to Teo et al. (2010) for more details. The overall procedure of BMRM is summarized in Algorithm 1. Under Assumption **A1**, Teo et al. (2010) showed that BMRM converges at $O(\frac{1}{\lambda\varepsilon})$ rates:

Theorem 3 *Suppose Assumption 1 holds. Then for any* $\varepsilon < 4G^2/\lambda$, BMRM *converges to an* ε *accurate solution of* (1) *as measured by* (9) *after at most the following number of steps:*

$$\log_2 \frac{\lambda J(\mathbf{0})}{G^2} + \frac{8G^2}{\lambda\varepsilon} - 1.$$

2.1 Lower Bound Preliminaries

Since most rates of convergence found in the literature are upper bounds, it is important to rigorously define the meaning of a lower bound with respect to ε, and to study its relationship with the upper bounds. At this juncture it is also important to clarify an important technical point. Instead of

Algorithm 1: BMRM.

Input: A tolerance level of function value ε.

Initialize: $\boldsymbol{w}_0$ to arbitrary value, for example, $\mathbf{0}$. Let $k = 1$.

while TRUE do

  Pick arbitrary subgradient of R_{emp} at $\mathbf{w}_{k-1}$: $\mathbf{a}_k \in \partial R_{\text{emp}}(\mathbf{w}_{k-1})$.

  Let $b_k = R_{\text{emp}}(\mathbf{w}_{k-1}) - \langle \mathbf{w}_{k-1}, \mathbf{a}_k \rangle$.

  Solve $\mathbf{w}_k = \operatorname{argmin}_{\mathbf{w}} J_k(\mathbf{w}) = \operatorname{argmin}_{\mathbf{w}} \left\{ \frac{\lambda}{2}\|\mathbf{w}\|^2 + \max_{i\in[k]} \{\langle \mathbf{w}, \mathbf{a}_i \rangle + b_i\} \right\}$.

  if $\min_{0\le t\le k} J(\mathbf{w}_t) - J_k(\mathbf{w}_k) < \varepsilon$ **then return** $\boldsymbol{w}_k$. **else** $k = k+1$.

minimizing the objective function $J(\mathbf{w})$ defined in (1), if we minimize a scaled version $cJ(\mathbf{w})$ this scales the approximation gap (9) by c. Assumption **A1** fixes this degree of freedom by bounding the scale of the objective function.

Given a function $f \in \mathcal{F}$ and an optimization algorithm A, suppose $\{\mathbf{w}_k\}$ are the iterates produced by some algorithm A when minimizing some function f. Define $T(\varepsilon; f, A)$ as the first step index k when $\mathbf{w}_k$ becomes an ε accurate solution:[4]

$$T(\varepsilon; f, A) = \min \{k : f(\mathbf{w}_k) - \min_{\boldsymbol{w}} f(\boldsymbol{w}) \le \varepsilon\}.$$

Upper and lower bounds are both properties for a pair of $\mathcal{F}$ and A. A function $\overline{\kappa}(\varepsilon)$ is called *an* upper bound of $(\mathcal{F}, A)$ if for all functions $f \in \mathcal{F}$ and all $\varepsilon > 0$, it takes *at most* order $\overline{\kappa}(\varepsilon)$ steps for A to reduce the gap to less than ε, that is,

$$\textbf{(UB)} \qquad \forall\, \varepsilon > 0, \forall\, f \in \mathcal{F},\ T(\varepsilon; f, A) \le \overline{\kappa}(\varepsilon).$$

On the other hand, we define lower bounds as follows. $\underline{\kappa}(\varepsilon)$ is called *a* lower bound of $(\mathcal{F}, A)$ if for any $\varepsilon > 0$, there exists a function $f_\varepsilon \in \mathcal{F}$ depending on ε, such that it takes *at least* $\underline{\kappa}(\varepsilon)$ steps for A to find an ε accurate solution of f_ε:

$$\textbf{(LB)} \qquad \forall\, \varepsilon > 0, \exists\, f_\varepsilon \in \mathcal{F},\ s.t.\ T(\varepsilon; f_\varepsilon, A) \ge \underline{\kappa}(\varepsilon).$$

Clearly, this lower bound is sufficient to refute upper bounds or to establish their tightness. The size of the function class $\mathcal{F}$ affects the upper and lower bounds in opposite ways. Suppose $\mathcal{F}' \subset \mathcal{F}$. Proving upper (resp. lower) bounds on $(\mathcal{F}', A)$ is usually easier (resp. harder) than proving upper (resp. lower) bounds for $(\mathcal{F}, A)$. Since the lower bounds studied by Nesterov (2003) and Nemirovski and Yudin (1983) are constructed by using *general* convex functions, they do not apply here to our specialized regularized risk objectives.

We are interested in bounding the *primal gap* of the iterates $\mathbf{w}_k$: $J(\mathbf{w}_k) - \min_{\boldsymbol{w}} J(\boldsymbol{w})$. Data sets will be constructed explicitly whose resulting objective $J(\mathbf{w})$ will be shown to satisfy Assumption **A1** and attain the lower bound of BMRM. We will focus on the R_{emp} for both the PRBEP loss in (3) and the ROCArea loss in (5).

4. The initial point also matters, as in the best case one can just start from the optimal solution. Thus the quantity of interest is actually $T(\varepsilon; f, A) := \max_{\mathbf{w}_0} \min\{k : f(\mathbf{w}_k) - \min_{\boldsymbol{w}} f(\boldsymbol{w}) \le \varepsilon$, starting point being $\mathbf{w}_0\}$. However, without loss of generality we assume some pre-specified way of initialization.

2.2 Construction Of Lower Bound For PRBEP Problem

Given $\varepsilon > 0$, let $n = \lceil 1/\varepsilon \rceil + 1$ and construct a data set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ as follows. Set $c = \frac{\sqrt{2}}{4}$, and let $\boldsymbol{e}_i$ denote the i-th coordinate vector in $\mathbb{R}^{n+2}$ ($p = n+2$). Let the first example be the only positive example: $y_1 = +1$ and $\boldsymbol{x}_1 = cn\boldsymbol{e}_2$. The rest $n-1$ examples are all negative: $y_i = -1$ and $\boldsymbol{x}_i = -cn\boldsymbol{e}_3 - cn\boldsymbol{e}_{i+2}$ for $i \in [2,n]$. This data set concretizes the R_{emp} in (3) as follows:

$$R_{\text{emp}}(\boldsymbol{w}) = \max_{\boldsymbol{z} \in \{-1,1\}^n} \underbrace{\left[\Delta(\boldsymbol{z}, \boldsymbol{y}) + \frac{1}{n} \langle \boldsymbol{w}, \boldsymbol{x}_1 \rangle (z_1 - 1) + \frac{1}{n} \sum_{i=2}^n \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle (z_i + 1) \right]}_{:= \Upsilon(\boldsymbol{z})}.$$

Setting $\lambda = 1$, we obtain the overall objective $J(\boldsymbol{w})$ in (1). Now we have the following lower bound on the rates of convergence of BMRM.

Theorem 4 *Let $\mathbf{w}_0 = \boldsymbol{e}_1 + 2c\boldsymbol{e}_2 + 2c\boldsymbol{e}_3$. Suppose running* BMRM *on the above constructed objective $J(\boldsymbol{w})$ produces iterates $\mathbf{w}_1, \ldots, \mathbf{w}_k, \ldots$. Then it takes* BMRM *at least $\left\lfloor \frac{1}{5\varepsilon} \right\rfloor$ steps to find an ε accurate solution. Formally,*

$$\min_{i \in [k]} J(\mathbf{w}_i) - \min_{\boldsymbol{w}} J(\boldsymbol{w}) \geq \frac{1}{4} \left(\frac{1}{k} - \frac{1}{n-1} \right) \text{ for all } k \in [n-1],$$

$$\text{hence} \quad \min_{i \in [k]} J(\mathbf{w}_i) - \min_{\boldsymbol{w}} J(\boldsymbol{w}) > \varepsilon \text{ for all } k < \frac{1}{5\varepsilon}.$$

Proof The crux of our proof is to show

$$\boldsymbol{w}_k = 2c\boldsymbol{e}_2 + 2c\boldsymbol{e}_3 + \frac{2c}{k} \sum_{i=4}^{k+3} \boldsymbol{e}_i, \quad \text{for all } k \in [n-1]. \tag{10}$$

We prove it by induction. Initially at $k = 1$, we note

$$\frac{1}{n} \langle \boldsymbol{w}_0, \boldsymbol{x}_1 \rangle = 2c^2 = \frac{1}{4}, \quad \text{and} \quad \frac{1}{n} \langle \boldsymbol{w}_0, \boldsymbol{x}_i \rangle = -2c^2 = -\frac{1}{4}, \quad \forall\, i \in [2,n]. \tag{11}$$

Since there is only one positive example, $\boldsymbol{z}$ can only have two cases. If $\boldsymbol{z} = \boldsymbol{y}$ (i.e., correct labeling with $b = 0$), then $\Upsilon(\boldsymbol{z}) = \Upsilon(\boldsymbol{y}) = 0$. If $\boldsymbol{z}$ misclassifies the only positive example into negative (i.e., $b = 1$), then $\Delta(\boldsymbol{z}, \boldsymbol{y}) = \frac{b}{n_+} = 1$. Since PRBEP forces $c = b = 1$, there must be one negative example misclassified into positive by $\boldsymbol{z}$. By (11), all $i \in [2,n]$ play the same role in $\Upsilon(\boldsymbol{z})$. Without loss of generality, assume the misclassified negative example is $i = 2$, that is, $\boldsymbol{z} = (-1, 1, -1, -1, \ldots)$. It is easy to check that $\Upsilon(\boldsymbol{z}) = 1 + \frac{2}{n}[-\langle \boldsymbol{w}_0, \boldsymbol{x}_1 \rangle + \langle \boldsymbol{w}_0, \boldsymbol{x}_2 \rangle] = 0 = \Upsilon(\boldsymbol{y})$, which means such $\boldsymbol{z}$ is a maximizer of $\Upsilon(\boldsymbol{z})$. Using this $\boldsymbol{z}$, we can derive a subgradient of $R_{\text{emp}}(\boldsymbol{w}_0)$:

$$\boldsymbol{a}_1 = \frac{2}{n}(-\boldsymbol{x}_1 + \boldsymbol{x}_2) = -2c\boldsymbol{e}_2 - 2c\boldsymbol{e}_3 - 2c\boldsymbol{e}_4 \text{ and } b_1 = 0 - \langle \boldsymbol{a}_1, \boldsymbol{w}_0 \rangle = 4c^2 + 4c^2 = 1,$$

$$\boldsymbol{w}_1 = \operatorname*{argmin}_{\boldsymbol{w}} \left\{ \frac{1}{2} \|\boldsymbol{w}\|^2 + \langle \boldsymbol{a}_1, \boldsymbol{w} \rangle + b_1 \right\} = 2c\boldsymbol{e}_2 + 2c\boldsymbol{e}_3 + 2c\boldsymbol{e}_4.$$

Next assume (10) holds for steps $1, \ldots, k$ ($k \in [n-2]$). Then at step $k+1$ we have

$$\frac{1}{n} \langle \boldsymbol{w}_k, \boldsymbol{x}_1 \rangle = 2c^2 = \frac{1}{4}, \quad \text{and} \quad \frac{1}{n} \langle \boldsymbol{w}_k, \boldsymbol{x}_i \rangle = \begin{cases} -2c^2 - \frac{2c^2}{k} = -\frac{1}{4} - \frac{1}{4k} & \forall\, i \in [2, k+1] \\ -2c^2 = -\frac{1}{4}, & \forall\, i \geq k+2 \end{cases}.$$

Again consider two cases of $\boldsymbol{z}$. If $\boldsymbol{z} = \boldsymbol{y}$ then $\Upsilon(\boldsymbol{z}) = \Upsilon(\boldsymbol{y}) = 0$. If $\boldsymbol{z}$ misclassifies the only positive example into negative, then $\Delta(\boldsymbol{z}, \boldsymbol{y}) = 1$. If $\boldsymbol{z}$ misclassifies any of $i \in [2, k+1]$ into positive, then $\Upsilon(\boldsymbol{z}) = 1 + \frac{2}{n}[-\langle \boldsymbol{w}_k, \boldsymbol{x}_1\rangle + \langle \boldsymbol{w}_k, \boldsymbol{x}_i\rangle] = -\frac{1}{2k} < \Upsilon(\boldsymbol{y})$. So such $\boldsymbol{z}$ cannot be a maximizer of $\Upsilon(\boldsymbol{z})$. But if $\boldsymbol{z}$ misclassifies any of $i \geq k+2$ into positive, then $\Upsilon(\boldsymbol{z}) = 1 + \frac{2}{n}[-\langle \boldsymbol{w}_k, \boldsymbol{x}_1\rangle + \langle \boldsymbol{w}_k, \boldsymbol{x}_i\rangle] = 0 = \Upsilon(\boldsymbol{y})$. So such $\boldsymbol{z}$ is a maximizer. Pick $i = k+2$ and we can derive from this $\boldsymbol{z}$ a subgradient of $R_{\text{emp}}(\boldsymbol{w}_k)$:

$$\boldsymbol{a}_{k+1} = \frac{2}{n}(-\boldsymbol{x}_1 + \boldsymbol{x}_{k+2}) = -2c\boldsymbol{e}_2 - 2c\boldsymbol{e}_3 - 2c\boldsymbol{e}_{k+4}, \quad b_{k+1} = 0 - \langle \boldsymbol{a}_{k+1}, \boldsymbol{w}_k\rangle = 4c^2 + 4c^2 = 1,$$

$$\boldsymbol{w}_{k+1} = \operatorname*{argmin}_{\boldsymbol{w}} J_{k+1}(\boldsymbol{w}) = \operatorname*{argmin}_{\boldsymbol{w}} \left\{ \frac{1}{2}\|\boldsymbol{w}\|^2 + \max_{i\in[k+1]} \{\langle \boldsymbol{a}_i, \boldsymbol{w}\rangle + b_i\} \right\} = 2c\boldsymbol{e}_2 + 2c\boldsymbol{e}_3 + \frac{2c}{k+1}\sum_{i=4}^{k+4} \boldsymbol{e}_i.$$

To verify the last step, note $\langle \boldsymbol{a}_i, \boldsymbol{w}_{k+1}\rangle + b_i = \frac{-1}{2(k+1)}$ for all $i \in [k+1]$, and so $\partial J_{k+1}(\boldsymbol{w}_{k+1}) = \{\boldsymbol{w}_{k+1} + \sum_{i=1}^{k+1} \alpha_i \boldsymbol{a}_i : \boldsymbol{\alpha} \in \Delta_{k+1}\} \ni \boldsymbol{0}$ (just set all $\alpha_i = \frac{1}{k+1}$). So (10) holds for step $k+1$. (End of induction.)

All that remains is to observe $J(\boldsymbol{w}_k) = \frac{1}{4}(2 + \frac{1}{k})$ while $\min_{\boldsymbol{w}} J(\boldsymbol{w}) \leq J(\boldsymbol{w}_{n-1}) = \frac{1}{4}(2 + \frac{1}{n-1})$, from which it follows that $J(\boldsymbol{w}_k) - \min_{\boldsymbol{w}} J(\boldsymbol{w}) \geq \frac{1}{4}(\frac{1}{k} - \frac{1}{n-1})$. ■

Note in the above run of BMRM, all subgradients $\boldsymbol{a}_k$ have norm $\sqrt{\frac{3}{2}}$ which satisfies Assumption 1.

2.3 Construction Of Lower Bound For ROCArea Problem

Next we consider the case of ROCArea. Given $\varepsilon > 0$, define $n = \lfloor 1/\varepsilon \rfloor$ and construct a data set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n+1}$ with $n+1$ examples as follows. Let $\boldsymbol{e}_i$ be the i-th coordinate vector in $\mathbb{R}^{n+2}$ ($p = n+2$). The first example is the only positive example with $y_1 = +1$ and $\boldsymbol{x}_1 = \sqrt{n}\boldsymbol{e}_2$. The rest n examples are all negative, with $y_i = -1$ and $\boldsymbol{x}_i = -n\boldsymbol{e}_{i+1}$ for $i \in [2, n+1]$. Then the corresponding empirical risk for ROCArea loss is

$$R_{\text{emp}}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=2}^{n+1} \max\{0, 1 - \langle \boldsymbol{w}, \boldsymbol{x}_1 - \boldsymbol{x}_i\rangle\} = \frac{1}{n}\sum_{i=2}^{n+1} \max\{0, 1 - \langle \boldsymbol{w}, \underbrace{\sqrt{n}\boldsymbol{e}_2 + n\boldsymbol{e}_{i+1}}_{:=\boldsymbol{u}_i}\rangle\}.$$

By further setting $\lambda = 1$, we obtain the overall objective $J(\boldsymbol{w})$ in (1). It is easy to see that the minimizer of $J(\boldsymbol{w})$ is $\mathbf{w}^* = \frac{1}{2}\left(\frac{1}{\sqrt{n}}\boldsymbol{e}_2 + \frac{1}{n}\sum_{i=3}^{n+2} \boldsymbol{e}_i\right)$ and $J(\mathbf{w}^*) = \frac{1}{4n}$. In fact, simply check that $\langle \mathbf{w}^*, \boldsymbol{u}_i\rangle = 1$ for all i, so $\partial J(\mathbf{w}^*) = \left\{\mathbf{w}^* - \sum_{i=1}^{n} \alpha_i \left(\frac{1}{\sqrt{n}}\boldsymbol{e}_2 + \boldsymbol{e}_{i+2}\right) : \alpha_i \in [0,1]\right\}$, and setting all $\alpha_i = \frac{1}{2n}$ yields the subgradient $\boldsymbol{0}$. Then we have the following lower bound on the rates of convergence for BMRM.

Theorem 5 *Let* $\mathbf{w}_0 = \boldsymbol{e}_1 + \frac{1}{\sqrt{n}}\boldsymbol{e}_2$. *Suppose running* BMRM *on the above constructed objective* $J(\boldsymbol{w})$ *produces iterates* $\mathbf{w}_1, \ldots, \mathbf{w}_k, \ldots$. *Then it takes* BMRM *at least* $\lfloor \frac{2}{3\varepsilon} \rfloor$ *steps to find an* ε *accurate solution. Formally,*

$$\min_{i\in[k]} J(\mathbf{w}_i) - J(\mathbf{w}^*) = \frac{1}{2k} + \frac{1}{4n} \text{ for all } k \in [n], \text{hence } \min_{i\in[k]} J(\mathbf{w}_i) - J(\mathbf{w}^*) > \varepsilon \text{ for all } k < \frac{2}{3\varepsilon}.$$

Proof The crux of the proof is to show that

$$\boldsymbol{w}_k = \frac{1}{\sqrt{n}}\boldsymbol{e}_2 + \frac{1}{k}\sum_{i=3}^{k+2} \boldsymbol{e}_i, \quad \forall\, k \in [n]. \tag{12}$$

We prove (12) by induction. Initially at $k=1$, we note that $\langle \boldsymbol{w}_0, \boldsymbol{u}_i \rangle = 1$ for all $i \in [2, n+1]$. Hence we can derive a subgradient and $\boldsymbol{w}_1$:

$$\boldsymbol{a}_1 = -\frac{1}{n}\boldsymbol{u}_2 = -\frac{1}{\sqrt{n}}\boldsymbol{e}_2 - \boldsymbol{e}_3, \qquad b_1 = -\langle \boldsymbol{a}_1, \boldsymbol{w}_0 \rangle = \frac{1}{n},$$
$$\boldsymbol{w}_1 = \operatorname*{argmin}_{\boldsymbol{w}} \left\{ \frac{1}{2}\|\boldsymbol{w}\|^2 + \langle \boldsymbol{a}_1, \boldsymbol{w} \rangle + b_1 \right\} = -\boldsymbol{a}_1 = \frac{1}{\sqrt{n}}\boldsymbol{e}_2 + \boldsymbol{e}_3.$$

So (12) holds for $k=1$. Now suppose it hold for steps $1, \ldots, k$ ($k \in [n-1]$). Then for step $k+1$ we note

$$\langle \boldsymbol{w}_k, \boldsymbol{u}_i \rangle = \begin{cases} 1 + \frac{n}{k} & \text{if } i \leq k \\ 1 & \text{if } i \geq k+1 \end{cases}.$$

Therefore we can derive a subgradient and compute $\boldsymbol{w}_{k+1}$:

$$\boldsymbol{a}_{k+1} = -\frac{1}{n}\boldsymbol{u}_{k+2} = -\frac{1}{\sqrt{n}}\boldsymbol{e}_2 - \boldsymbol{e}_{k+3}, \qquad b_{k+1} = -\langle \boldsymbol{a}_{k+1}, \boldsymbol{w}_k \rangle = \frac{1}{n},$$
$$\boldsymbol{w}_{k+1} = \operatorname*{argmin}_{\boldsymbol{w}} \underbrace{\left\{ \frac{1}{2}\|\boldsymbol{w}\|^2 + \max_{i \in [k+1]} \{\langle \boldsymbol{a}_i, \boldsymbol{w} \rangle + b_i\} \right\}}_{:= J_{k+1}(\boldsymbol{w})} = -\frac{1}{k+1}\sum_{i=1}^{k+1} \boldsymbol{a}_i = \frac{1}{\sqrt{n}}\boldsymbol{e}_2 + \frac{1}{k+1}\sum_{i=3}^{k+3} \boldsymbol{e}_i.$$

This minimizer can be verified by noting that $\langle \boldsymbol{w}_{k+1}, \boldsymbol{a}_i \rangle + b_i = -\frac{1}{k+1}$ for all $i \in [k+1]$, and so $\partial J_{k+1}(\boldsymbol{w}_{k+1}) = \{\boldsymbol{w}_{k+1} + \sum_{i=1}^{k+1} \alpha_i \boldsymbol{a}_i : \boldsymbol{\alpha} \in \Delta_{k+1}\} \ni \boldsymbol{0}$ (just set all $\alpha_i = \frac{1}{k+1}$). So (12) holds for step $k+1$. (End of induction.)

All that remains is to observe $J(\boldsymbol{w}_k) = \frac{1}{2}(\frac{1}{n} + \frac{1}{k})$. ■

As before, in the above run of BMRM, all subgradients $\boldsymbol{a}_k$ have norm $(1+\frac{1}{n})^{1/2}$ which satisfies Assumption 1.

3. Reformulating The Empirical Risk

In order to bypass the lower bound of CPMs and show that the multivariate score problems can be optimized at faster rates, we propose a smoothing strategy as described in Section 1.1. In this section, we will focus on the design of smooth surrogate functions, with emphasis on the consequent rates of convergence when optimized by Nesterov's accelerated gradient methods (Nesterov, 2005). The computational issues will be tackled in Section 4.

In order to approximate R_{emp} by $g_\mu^\star$ we will start by writing $R_{\text{emp}}(\mathbf{w})$ as $g^*(A^\top \mathbf{w})$ for an appropriate linear transform A and convex function g. Let the domain of g be Q and d be a strongly convex function with modulus 1 defined on Q. Furthermore, assume $\min_{\boldsymbol{\alpha} \in Q} d(\boldsymbol{\alpha}) = 0$ and denote $D = \max_{\boldsymbol{\alpha} \in Q} d(\boldsymbol{\alpha})$. In optimization parlance d is called a *prox-function*. Set

$$g_\mu^\star = (g + \mu d)^\star.$$

Then by the analysis in Appendix A, we can conclude that $g_\mu^\star(A^\top \mathbf{w})$ has a Lipschitz continuous gradient with constant at most $\frac{1}{\mu}\|A\|^2$. In addition,

$$\left| g_\mu^\star(A^\top \mathbf{w}) - R_{\text{emp}}(\mathbf{w}) \right| \leq \mu D. \tag{13}$$

Choosing $\mu = \varepsilon/D$, we can guarantee the approximation is uniformly upper bounded by ε.

There are indeed many different ways of expressing $R_{\text{emp}}(\mathbf{w})$ as $g^\star(A^\top \mathbf{w})$, but the next two sections will demonstrate the advantage of our design.

3.1 Contingency Table Based Loss

Letting $\mathcal{S}^k$ denote the k dimensional probability simplex, we can rewrite (3) as:

$$R_{\text{emp}}(\mathbf{w}) = \max_{\boldsymbol{z}\in\{-1,1\}^n} \left[\Delta(\boldsymbol{z},\boldsymbol{y}) + \frac{1}{n}\sum_{i=1}^n \langle \mathbf{w}, \boldsymbol{x}_i\rangle (z_i - y_i)\right] \tag{14}$$

$$= \max_{\boldsymbol{\alpha}\in\mathcal{S}^{2^n}} \sum_{\boldsymbol{z}\in\{-1,1\}^n} \alpha_{\boldsymbol{z}} \left(\Delta(\boldsymbol{z},\boldsymbol{y}) + \frac{1}{n}\sum_{i=1}^n \langle \mathbf{w}, \boldsymbol{x}_i\rangle (z_i - y_i)\right) \tag{15}$$

$$= \max_{\boldsymbol{\alpha}\in\mathcal{S}^{2^n}} \sum_{\boldsymbol{z}\in\{-1,1\}^n} \alpha_{\boldsymbol{z}}\Delta(\boldsymbol{z},\boldsymbol{y}) - \frac{2}{n}\sum_{i=1}^n y_i \langle \mathbf{w}, \boldsymbol{x}_i\rangle \left(\sum_{\boldsymbol{z}:z_i=-y_i} \alpha_{\boldsymbol{z}}\right).$$

Note from (14) to (15), we defined $0\cdot(-\infty) = 0$ since $\Delta(\boldsymbol{z},\boldsymbol{y})$ can be $-\infty$ as in (4). Introduce $\beta_i = \sum_{\boldsymbol{z}:z_i=-y_i} \alpha_{\boldsymbol{z}}$. Then $\beta_i \in [0,1]$ and one can rewrite the above equation in terms of $\boldsymbol{\beta}$:

$$R_{\text{emp}}(\mathbf{w}) = \max_{\boldsymbol{\beta}\in[0,1]^n} \left\{\frac{-2}{n}\sum_{i=1}^n y_i \langle \mathbf{w}, \boldsymbol{x}_i\rangle \beta_i - g(\boldsymbol{\beta})\right\} \tag{16}$$

$$\text{where}\quad g(\boldsymbol{\beta}) := -\max_{\boldsymbol{\alpha}\in\mathcal{A}(\boldsymbol{\beta})} \sum_{\boldsymbol{z}\in\{-1,1\}^n} \alpha_{\boldsymbol{z}}\Delta(\boldsymbol{z},\boldsymbol{y}). \tag{17}$$

Here $\mathcal{A}(\boldsymbol{\beta})$ is a subset of $\mathcal{S}^{2^n}$ defined via $\mathcal{A}(\boldsymbol{\beta}) = \left\{\boldsymbol{\alpha} \text{ s.t. } \sum_{\boldsymbol{z}:z_i=-y_i} \alpha_{\boldsymbol{z}} = \beta_i \text{ for all } i\right\}$. Indeed, this rewriting only requires that the mapping from $\boldsymbol{\alpha}\in\mathcal{S}^{2^n}$ to $\boldsymbol{\beta}\in Q := [0,1]^n$ is surjective. This is clear because for any $\boldsymbol{\beta}\in[0,1]^n$, a pre-image $\boldsymbol{\alpha}$ can be constructed by setting:

$$\alpha_{\boldsymbol{z}} = \prod_{i=1}^n \gamma_i, \quad \text{where} \quad \gamma_i = \begin{cases} \beta_i & \text{if } z_i = -y_i \\ 1-\beta_i & \text{if } z_i = y_i \end{cases}.$$

Proposition 6 *$g(\boldsymbol{\beta})$ defined by (17) is convex on $\boldsymbol{\beta}\in[0,1]^n$.*

Proof Clearly $\mathcal{A}(\boldsymbol{\beta})$ is closed for any $\boldsymbol{\beta}\in Q$. So without loss of generality assume that for any $\boldsymbol{\beta}$ and $\boldsymbol{\beta}'$ in $[0,1]^n$, the maximum in (17) is attained at $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ respectively. So $\sum_{\boldsymbol{z}:z_i=-y_i} \alpha_{\boldsymbol{z}} = \beta_i$ and $\sum_{\boldsymbol{z}:z_i=-y_i} \alpha'_{\boldsymbol{z}} = \beta'_i$. For any $\rho\in[0,1]$, consider $\boldsymbol{\beta}^\rho := \rho\boldsymbol{\beta} + (1-\rho)\boldsymbol{\beta}'$. Define $\boldsymbol{\alpha}^\rho := \rho\boldsymbol{\alpha} + (1-\rho)\boldsymbol{\alpha}'$. Then clearly $\boldsymbol{\alpha}^\rho$ satisfies

$$\sum_{\boldsymbol{z}:z_i=-y_i} \alpha^\rho_{\boldsymbol{z}} = \rho\beta_i + (1-\rho)\beta'_i = \beta^\rho_i.$$

Therefore $\boldsymbol{\alpha}^\rho$ is admissible in the definition of $g(\boldsymbol{\beta}^\rho)$ (17), and

$$g(\boldsymbol{\beta}^\rho) \le -\sum_{\boldsymbol{z}} \alpha^\rho_{\boldsymbol{z}}\Delta(\boldsymbol{z},\boldsymbol{y}) = -\rho\sum_{\boldsymbol{z}} \alpha_{\boldsymbol{z}}\Delta(\boldsymbol{z},\boldsymbol{y}) - (1-\rho)\sum_{\boldsymbol{z}} \alpha'_{\boldsymbol{z}}\Delta(\boldsymbol{z},\boldsymbol{y})$$
$$= \rho g(\boldsymbol{\beta}) + (1-\rho) g(\boldsymbol{\beta}').$$

which establishes the convexity of g. ■

Using (16) it immediately follows that $R_{\text{emp}}(\mathbf{w}) = g^\star(A^\top\mathbf{w})$ where A is a p-by-n matrix whose i-th column is $\frac{-2}{n} y_i \boldsymbol{x}_i$,.

3.1.1 $\sqrt{D}\,\|A\|$ FOR OUR DESIGN

Let us choose the prox-function $d(\boldsymbol{\beta})$ as $\frac{1}{2}\|\boldsymbol{\beta}\|^2$. Then $D = \max_{\boldsymbol{\beta}\in[0,1]^n} d(\boldsymbol{\beta}) = \frac{n}{2}$. Define $A = \frac{-2}{n}(y_1\boldsymbol{x}_1,\ldots,y_n\boldsymbol{x}_n)$ and M be a n-by-n matrix with $M_{ij} = y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle$. Clearly, $A^\top A = \frac{4}{n^2}M$. Because we assumed that $\|\boldsymbol{x}_i\| \le R$, it follows that $|M_{ij}| \le R^2$. The norm of A can be upper bounded as follows:

$$\|A\|^2 = \lambda_{\max}(A^\top A) = \frac{4}{n^2}\lambda_{\max}(M) \le \frac{4}{n^2}\operatorname{tr}(M) \le \frac{4}{n^2}\cdot nR^2 = \frac{4R^2}{n}. \tag{18}$$

Hence

$$\sqrt{D}\,\|A\| \le \sqrt{\frac{n}{2}}\frac{2R}{\sqrt{n}} = \sqrt{2}R.$$

This constant upper bound is highly desirable because it implies that the rate of convergence in (7) is independent of the size of the data set (n or p). However, the cost per iteration still depends on n and p.

3.1.2 ALTERNATIVES

It is illuminating to see how naive choices for smoothing R_{emp} can lead to excessively large values of $\sqrt{D}\,\|A\|$. For instance, using (14), $R_{\text{emp}}(\mathbf{w})$ can be rewritten as

$$\begin{aligned} R_{\text{emp}}(\mathbf{w}) &= \max_{\boldsymbol{z}\in\{-1,1\}^n} \frac{1}{n}\left[n\Delta(\boldsymbol{z},\boldsymbol{y}) + \sum_{i=1}^n \langle \mathbf{w}, \boldsymbol{x}_i\rangle (z_i - y_i)\right] \\ &= \max_{\boldsymbol{\alpha}\in Q} \sum_{\boldsymbol{z}\in\{-1,1\}^n} \alpha_{\boldsymbol{z}}\left(n\Delta(\boldsymbol{z},\boldsymbol{y}) + \sum_{i=1}^n \langle \mathbf{w}, \boldsymbol{x}_i\rangle (z_i - y_i)\right). \end{aligned}$$

where Q refers to the 2^n dimensional simplex scaled by $\frac{1}{n}$: $Q = \left\{\boldsymbol{\alpha} : \alpha_{\boldsymbol{z}} \in [0, n^{-1}] \text{ and } \sum_{\boldsymbol{z}} \alpha_{\boldsymbol{z}} = \frac{1}{n}\right\}$. Defining A as a p-by-2^n matrix whose $\boldsymbol{z}$-th column is given by $\sum_{i=1}^n \boldsymbol{x}_i(z_i - y_i)$ and writing

$$h(\boldsymbol{\alpha}) = \begin{cases} -n\sum_{\boldsymbol{z}\in\{-1,1\}^n}\Delta(\boldsymbol{z},\boldsymbol{y})\alpha_{\boldsymbol{z}} & \text{if } \boldsymbol{\alpha}\in Q \\ +\infty & \text{otherwise} \end{cases},$$

we can express R_{emp} as

$$R_{\text{emp}}(\mathbf{w}) = \max_{\boldsymbol{\alpha}\in Q}\left[\left\langle A^\top\mathbf{w}, \boldsymbol{\alpha}\right\rangle - h(\boldsymbol{\alpha})\right] = h^\star(A^\top\mathbf{w}). \tag{19}$$

Now let us investigate the value of $\sqrt{D}\,\|A\|$. As Q is a scaled simplex, a natural choice of prox-function is

$$d(\boldsymbol{\alpha}) = \sum_{\boldsymbol{z}} \alpha_{\boldsymbol{z}}\log\alpha_{\boldsymbol{z}} + \frac{1}{n}\log n + \log 2.$$

Clearly, $d(\boldsymbol{\alpha})$ is a strongly convex function in Q with modulus of strong convexity 1 under the L_1 norm (see, e.g., Beck and Teboulle, 2003, Proposition 5.1). In addition, $d(\boldsymbol{\alpha})$ is minimized when $\boldsymbol{\alpha}$ is uniform, that is, $\alpha_{\boldsymbol{z}} = \frac{1}{n2^n}$ for all $\boldsymbol{z}$, and so

$$\min_{\boldsymbol{\alpha}\in Q} d(\boldsymbol{\alpha}) = 2^n\frac{1}{n2^n}\log\frac{1}{n2^n} + \frac{1}{n}\log n + \log 2 = 0.$$

Therefore $d(\boldsymbol{\alpha})$ satisfies all the prerequisites of being a prox-function. Furthermore, $d(\boldsymbol{\alpha})$ is maximized on the corner of the scaled simplex, for example, $\alpha_{\boldsymbol{y}} = \frac{1}{n}$ and $\alpha_{\boldsymbol{z}} = 0$ for all $\boldsymbol{z} \neq \boldsymbol{y}$. So

$$D = \max_{\boldsymbol{\alpha} \in Q} d(\boldsymbol{\alpha}) = \frac{1}{n} \log \frac{1}{n} + \frac{1}{n} \log n + \log 2 = \log 2. \tag{20}$$

Finally, denoting the $\mathbf{z}$-th column of A as $A_{:\boldsymbol{z}}$, we can compute $\|A\|$ as

$$\begin{aligned} \|A\| &= \max_{\mathbf{u}:\|\mathbf{u}\|=1, \mathbf{v}:\sum_{\boldsymbol{z}}|v_{\boldsymbol{z}}|=1} \mathbf{u}^\top A \mathbf{v} = \max_{\mathbf{v}:\sum_{\boldsymbol{z}}|v_{\boldsymbol{z}}|=1} \|A\mathbf{v}\| = \max_{\mathbf{z}} \|A_{:\mathbf{z}}\| \\ &= \max_{\mathbf{z}} \left\| \sum_{i=1}^{n} \boldsymbol{x}_i (z_i - y_i) \right\| \leq 2 \sum_{i=1}^{n} \|\boldsymbol{x}_i\| \leq 2nR. \end{aligned} \tag{21}$$

Note this bound is tight because $\|A\| = 2nR$ can be attained by setting $\boldsymbol{x}_i = R\boldsymbol{e}_1$ for all i, in which case the maximizing $\boldsymbol{z}$ is simply $-\boldsymbol{y}$. Eventually combining (20) and (21), we arrive at

$$\sqrt{D}\,\|A\| \leq 2nR\sqrt{\log 2}.$$

Thus if we re-express R_{emp} by using this alternative form as in (19), $\sqrt{D}\,\|A\|$ will scale as $\Theta(nR)$ which grows linearly with n, the number of training examples.

3.2 ROCArea Loss

We rewrite $R_{\text{emp}}(\mathbf{w})$ from (5) as:

$$\frac{1}{m} \max_{z_{ij} \in \{0,1\}} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} z_{ij} [1 - \mathbf{w}^\top (\mathbf{x}_i - \mathbf{x}_j)] = \frac{1}{m} \max_{\boldsymbol{\beta} \in [0,1]^m} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} \beta_{ij} \left[1 - \boldsymbol{w}^\top (\boldsymbol{x}_i - \boldsymbol{x}_j) \right]. \tag{22}$$

The above empirical risk can be identified with $g^\star(A^\top \mathbf{w})$ by setting

$$g(\boldsymbol{\beta}) = \begin{cases} -\frac{1}{m} \sum_{i,j} \beta_{ij} & \text{if } \boldsymbol{\beta} \in [0,1]^m \\ +\infty & \text{elsewhere} \end{cases},$$

and letting A be a p-by-m matrix whose (ij)-th column is $-\frac{1}{m}(\boldsymbol{x}_i - \boldsymbol{x}_j)$. This g is a linear function defined over a convex domain, and hence clearly convex.

3.2.1 $\sqrt{D}\,\|A\|$ FOR OUR DESIGN

Choose prox-function $d(\boldsymbol{\beta}) = \frac{1}{2} \sum_{i,j} \beta_{ij}^2$. By a simple calculation, $D = \max_{\boldsymbol{\beta}} d(\boldsymbol{\beta}) = \frac{m}{2}$. As before, define A to be a matrix such that its (ij)-th column $A_{ij} = -\frac{1}{m}(\boldsymbol{x}_i - \boldsymbol{x}_j)$ and M be a m-by- m matrix with $M_{(ij)(i'j')} = \langle \boldsymbol{x}_i - \boldsymbol{x}_j, \boldsymbol{x}_{i'} - \boldsymbol{x}_{j'} \rangle$. Clearly, $A^\top A = \frac{1}{m^2} M$. Because we assumed that $\|\boldsymbol{x}_i\| \leq R$, it follows that $|M_{ij}| \leq 4R^2$. The norm of A can be upper bounded as follows:

$$\|A\|^2 = \lambda_{\max}(A^\top A) = \frac{1}{m^2} \lambda_{\max}(M) \leq \frac{1}{m^2} \operatorname{tr}(M) \leq \frac{1}{m^2} \cdot 4mR^2 = \frac{4R^2}{m}.$$

Therefore

$$\sqrt{D}\,\|A\| \leq \sqrt{\frac{m}{2}} \cdot \frac{2R}{\sqrt{m}} = \sqrt{2}R. \tag{23}$$

This upper bound is independent of n and guarantees that the rate of convergence is not affected by the size of the data set.

3.2.2 ALTERNATIVES

The advantage of our above design of $g^\star(A^\top \boldsymbol{w})$ can be seen by comparing it with an alternative design. In (22), let us introduce

$$\gamma_i = \sum_{j \in \mathcal{N}} \beta_{ij},\ i \in \mathcal{P}, \qquad \text{and} \qquad \gamma_j = \sum_{i \in \mathcal{P}} \beta_{ij},\ j \in \mathcal{N}.$$

Then clearly $\boldsymbol{\gamma} \in \mathcal{B} := \{\gamma_i \in [0, n_-], \gamma_j \in [0, n_+]\}$. However this map from $\boldsymbol{\beta} \in [0,1]^m$ to $\boldsymbol{\gamma} \in \mathcal{B}$ is obviously *not* surjective, because the image $\boldsymbol{\gamma}$ should at least satisfy $\sum_{i \in \mathcal{P}} \gamma_i = \sum_{j \in \mathcal{N}} \gamma_j$.[5] So let us denote the real range of the mapping as $\mathcal{A}$ and $\mathcal{A} \subseteq \mathcal{B}$. Now rewrite (22) in terms of $\boldsymbol{\gamma}$:

$$\begin{aligned} R_{\text{emp}}(\boldsymbol{w}) &= \frac{1}{m} \max_{\boldsymbol{\gamma} \in \mathcal{A}} \left\{ \sum_{i \in \mathcal{P}} \gamma_i \boldsymbol{w}^\top (-\boldsymbol{x}_i) + \sum_{j \in \mathcal{N}} \gamma_j \boldsymbol{w}^\top \boldsymbol{x}_j + \sum_{i \in \mathcal{P}} \gamma_i + \sum_{j \in \mathcal{N}} \gamma_j \right\} \\ &= g^\star(A^\top \boldsymbol{w}), \quad \text{where} \quad g(\boldsymbol{\gamma}) = \begin{cases} -\frac{1}{m} \sum_{i \in \mathcal{P}} \gamma_i + \frac{1}{m} \sum_{j \in \mathcal{N}} \gamma_j & \text{if } \boldsymbol{\gamma} \in \mathcal{A} \\ +\infty & \text{elsewhere} \end{cases}, \end{aligned}$$

and A is a p-by-n matrix whose i-th column is $-\frac{1}{m}\boldsymbol{x}_i$ ($i \in \mathcal{P}$) and j-th column is $\frac{1}{m}\boldsymbol{x}_j$ ($j \in \mathcal{N}$).

Choose a prox-function $d(\boldsymbol{\gamma}) = \frac{1}{2}\|\boldsymbol{\gamma}\|^2$. Then we can bound $\|A\|$ by $\frac{R}{m}\sqrt{n}$ similar to (18) and bound D by

$$D = \max_{\boldsymbol{\gamma} \in \mathcal{A}} d(\boldsymbol{\gamma}) \le \max_{\boldsymbol{\gamma} \in \mathcal{B}} d(\boldsymbol{\gamma}) = \frac{1}{2}\left(n_+^2 n_- + n_+ n_-^2\right).$$

Note that the $\le$ here is actually an equality because they have a common maximizer: $\gamma_i = n_-$ for all $i \in \mathcal{P}$ and $\gamma_j = n_+$ for all $j \in \mathcal{N}$ (which is mapped from all $\beta_{ij} = 1$). Now we have

$$\sqrt{D}\,\|A\| \le \frac{1}{\sqrt{2}} \sqrt{n_+^2 n_- + n_+ n_-^2}\, \frac{R}{m}\sqrt{n} = \frac{R}{\sqrt{2}} \sqrt{\frac{(n_+ + n_-)^2}{n_+ n_-}}.$$

This bound depends on the ratio between positive and negative examples. If the class is balanced and there exist constants $c_+, c_- \in (0,1)$ such that $n_+ \ge c_+ n$ and $n_- \ge c_- n$, then we easily derive $\sqrt{D}\,\|A\| \le \frac{R}{\sqrt{2c_+c_-}}$ which is a constant. However, if we fix $n_- = 1$ and $n_+ = n-1$, then $\sqrt{D}\,\|A\|$ will scale as $\Theta(\sqrt{n})$. In contrast, the bound in (23) is always a constant.

4. Efficient Evaluation Of The Smoothed Objective And Its Gradient

The last building block required to make our smoothing scheme work is an efficient algorithm to compute the smoothed empirical risk $g_\mu^\star(A^\top \mathbf{w})$ and its gradient. By the chain rule and Corollary X.1.4.4 of Hiriart-Urruty and Lemaréchal (1996), we have

$$\frac{\partial}{\partial \mathbf{w}} g_\mu^\star(A^\top \mathbf{w}) = A\hat{\boldsymbol{\beta}} \quad \text{where} \quad \hat{\boldsymbol{\beta}} = \operatorname*{argmax}_{\boldsymbol{\beta} \in \mathcal{Q}} \left\langle \boldsymbol{\beta}, A^\top \mathbf{w} \right\rangle - g(\boldsymbol{\beta}) - \mu d(\boldsymbol{\beta}). \tag{24}$$

Since g is convex and d is strongly convex, the optimal $\hat{\boldsymbol{\beta}}$ must be unique. Two major difficulties arise in computing the above gradient: the optimal $\hat{\boldsymbol{\beta}}$ can be hard to solve (e.g., in the case of contingency table based loss), and the matrix vector product in (24) can be costly (e.g., $O(n^2 p)$ for ROCArea). Below we show how these difficulties can be overcome to compute the gradient in $O(n \log n)$ time.

5. Note that even adding this equality constraint to the definition of $\mathcal{B}$ will still not make the map surjective.

4.1 Contingency Table Based Loss

Since A is a $p \times n$ dimensional matrix and $\hat{\boldsymbol{\beta}}$ is a n dimensional vector, the matrix vector product in (24) can be computed in $O(np)$ time. Below we focus on solving $\hat{\boldsymbol{\beta}}$ in (24).

To take into account the constraints in the definition of $g(\boldsymbol{\beta})$, we introduce Lagrangian multipliers θ_i and the optimization in (24) becomes

$$\begin{aligned} g_\mu^\star(A^\top \mathbf{w}) = \max_{\boldsymbol{\beta}\in[0,1]^n} \Bigg\{ & \frac{-2}{n}\sum_{i=1}^n y_i \langle \mathbf{w}, \boldsymbol{x}_i\rangle \beta_i - \frac{\mu}{2}\sum_{i=1}^n \beta_i^2 \\ & + \max_{\boldsymbol{\alpha}\in\mathcal{S}^{2^n}} \left[\sum_{\boldsymbol{z}} \alpha_{\boldsymbol{z}} \Delta(\boldsymbol{z},\boldsymbol{y}) + \min_{\boldsymbol{\theta}\in\mathbb{R}^n} \sum_{i=1}^n \theta_i \left(\sum_{\boldsymbol{z}: z_i = -y_i} \alpha_{\boldsymbol{z}} - \beta_i \right) \right] \Bigg\} \end{aligned} \tag{25}$$

$$\begin{aligned} \Leftrightarrow \min_{\boldsymbol{\theta}\in\mathbb{R}^n} \Bigg\{ & \max_{\boldsymbol{\alpha}\in\mathcal{S}^{2^n}} \sum_{\boldsymbol{z}} \alpha_{\boldsymbol{z}} \left[\Delta(\boldsymbol{z},\boldsymbol{y}) + \sum_i \theta_i \delta(z_i = -y_i) \right] \\ & + \max_{\boldsymbol{\beta}\in[0,1]^n} \sum_{i=1}^n \left(\frac{-\mu}{2}\beta_i^2 - \left(\frac{2}{n} y_i \langle \mathbf{w}, \boldsymbol{x}_i\rangle + \theta_i \right) \beta_i \right) \Bigg\} \end{aligned}$$

$$\Leftrightarrow \min_{\boldsymbol{\theta}\in\mathbb{R}^n} \Bigg\{ \max_{\boldsymbol{z}} \underbrace{\left[\Delta(\boldsymbol{z},\boldsymbol{y}) + \sum_i \theta_i \delta(z_i = -y_i) \right]}_{:= q(\boldsymbol{z},\boldsymbol{\theta})} + \sum_{i=1}^n \underbrace{\max_{\beta_i\in[0,1]} \left[\frac{-\mu}{2}\beta_i^2 - \left(\frac{2}{n} y_i \langle \mathbf{w}, \boldsymbol{x}_i\rangle + \theta_i \right) \beta_i \right]}_{:= h_i(\theta_i)} \Bigg\}. \tag{26}$$

The last step is because all β_i are decoupled and can be optimized independently. Let

$$D(\boldsymbol{\theta}) := \max_{\boldsymbol{z}} q(\boldsymbol{z},\boldsymbol{\theta}) + \sum_{i=1}^n h_i(\theta_i),$$

and $\hat{\boldsymbol{\theta}}$ be a minimizer of $D(\boldsymbol{\theta})$: $\hat{\boldsymbol{\theta}} \in \operatorname{argmin}_{\boldsymbol{\theta}} D(\boldsymbol{\theta})$. Given $\hat{\boldsymbol{\theta}}$ and denoting

$$a_i = \frac{-2}{n} y_i \langle \mathbf{w}, \boldsymbol{x}_i \rangle, \tag{27}$$

we can recover the optimal $\beta(\hat{\theta}_i)$ from the definition of $h_i(\hat{\theta}_i)$ as follows:

$$\hat{\beta}_i = \beta_i(\hat{\theta}_i) = \begin{cases} 0 & \text{if } \hat{\theta}_i \geq a_i \\ 1 & \text{if } \hat{\theta}_i \leq a_i - \mu \\ \frac{1}{\mu}(a_i - \hat{\theta}_i) & \text{if } \hat{\theta}_i \in [a_i - \mu, a_i] \end{cases}.$$

So, the main challenge that remains is to compute $\hat{\boldsymbol{\theta}}$. Towards this end, first note that:

$$\nabla_{\theta_i} h_i(\theta_i) = -\beta_i(\theta_i) \quad \text{and} \quad \partial_{\boldsymbol{\theta}} q(\boldsymbol{z},\boldsymbol{\theta}) = \operatorname{co}\left\{ \delta_{\boldsymbol{z}} : \boldsymbol{z} \in \operatorname*{argmax}_{\boldsymbol{z}} q(\boldsymbol{z},\boldsymbol{\theta}) \right\}.$$

Here $\delta_{\boldsymbol{z}} := (\delta(z_1 = -y_1), \ldots, \delta(z_n = -y_n))^\top$ and $\operatorname{co}(\cdot)$ denotes the convex hull of a set. By the first order optimality conditions $\mathbf{0} \in \partial D(\hat{\boldsymbol{\theta}})$, which implies that

$$\hat{\boldsymbol{\beta}} \in \operatorname{co}\left\{ \delta_{\boldsymbol{z}} : \boldsymbol{z} \in \operatorname*{argmax}_{\boldsymbol{z}} q(\boldsymbol{z},\boldsymbol{\theta}) \right\}. \tag{28}$$

We will henceforth restrict our attention to PRBEP as a special case of contingency table based loss Δ. A major contribution of this paper is a characterization of the optimal solution of (26), based on which we propose an $O(n \log n)$ complexity algorithm for finding $\hat{\boldsymbol{\theta}}$ exactly. To state the result, our notation needs to be refined. We will always use i to index positive examples ($i \in \mathcal{P}$) and j to index negative examples ($j \in \mathcal{N}$). Let the positive examples be associated with $\boldsymbol{x}_i^+$, $y_i^+ (= 1)$, a_i^+, $\hat{\theta}_i^+$, $\hat{\beta}_i^+$, and $h_i^+(\cdot)$, and the negative examples be associated with $\boldsymbol{x}_j^-$, $y_j^- (= -1)$, a_j^-, $\hat{\theta}_j^-$, $\hat{\beta}_j^-$, and $h_j^-(\cdot)$. Given two arbitrary scalars θ^+ and θ^-, let us define $\boldsymbol{\theta} = \boldsymbol{\theta}(\theta^+, \theta^-)$ by setting $\theta_i^+ = \theta^+$ for all $i \in \mathcal{P}$ and $\theta_j^- = \theta^-$ for all $j \in \mathcal{N}$. Finally define $f : \mathbb{R} \mapsto \mathbb{R}$ as follows

$$f(\theta) = \sum_{i \in \mathcal{P}} h_i^+(\theta) + \sum_{j \in \mathcal{N}} h_j^- \left(\frac{-1}{n_+} - \theta \right).$$

Then f is differentiable and convex in θ. Now our result can be stated as

Theorem 7 *The problem* (26) *must have an optimal solution which takes the form* $\boldsymbol{\theta}(\theta^+, \theta^-)$ *with* $\theta^+ + \theta^- = \frac{-1}{n_+}$. *Conversely, suppose* $\theta^+ + \theta^- = \frac{-1}{n_+}$, *then* $\boldsymbol{\theta}(\theta^+, \theta^-)$ *is an optimal solution if, and only if,* $\nabla f(\theta^+) = 0$.

Remark 8 *Although the optimal* $\boldsymbol{\theta}$ *to the problem* (26) *may be not unique, the optimal* $\boldsymbol{\beta}$ *to the problem* (25) *must be unique as we commented below* (24).

4.1.1 PROOF OF THEOREM 7

We start with the first sentence of Theorem 7. Our idea is as follows: given any optimal solution $\hat{\boldsymbol{\theta}}$, we will construct θ^+ and θ^- which satisfy $\theta^+ + \theta^- = \frac{-1}{n_+}$ and $D(\boldsymbol{\theta}(\theta^+, \theta^-)) \leq D(\hat{\boldsymbol{\theta}})$.

First $\hat{\boldsymbol{\theta}}$ must be in $\mathbb{R}^n$ because $D(\boldsymbol{\theta})$ tends to ∞ when any component of $\boldsymbol{\theta}$ approaches ∞ or $-\infty$. Without loss of generality, assume $n_+ \leq n_-$, $\hat{\theta}_1^+ \geq \hat{\theta}_2^+ \geq \ldots$, and $\hat{\theta}_1^- \geq \hat{\theta}_2^- \geq \ldots$. Define $\hat{\theta}_0^+ = \hat{\theta}_0^- = 0$, and consider the phase transition point

$$k := \max \left\{ 0 \leq i \leq n_+ : \hat{\theta}_i^+ + \hat{\theta}_i^- + \frac{1}{n_+} > 0 \right\},$$

based on which we have three cases: $k = 0$, $1 \leq k \leq n_+ - 1$, and $k = n_+$.

Case 1: $1 \leq k \leq n_+ - 1$. Then there must exist $\theta^+ \in [\hat{\theta}_{k+1}^+, \hat{\theta}_k^+]$ and $\theta^- \in [\hat{\theta}_{k+1}^-, \hat{\theta}_k^-]$ such that $\theta^+ + \theta^- = \frac{-1}{n_+}$. Denote $\boldsymbol{\theta}' := \boldsymbol{\theta}(\theta^+, \theta^-)$ and it suffices to show $D(\boldsymbol{\theta}') \leq D(\hat{\boldsymbol{\theta}})$. Since $\theta^+ + \theta^- = \frac{-1}{n_+}$, it is easy to see $\max_{\boldsymbol{z}} q(\boldsymbol{z}, \boldsymbol{\theta}') = 0$, and so

$$D(\boldsymbol{\theta}') = \sum_{i \in \mathcal{P}} h_i^+(\theta^+) + \sum_{j \in \mathcal{N}} h_j^-(\theta^-).$$

To get a lower bound of $D(\hat{\boldsymbol{\theta}})$, consider a specific $\boldsymbol{z}$ where $z_i^+ = -1$ for $1 \leq i \leq k$, $z_i^+ = 1$ for $i > k$, $z_j^- = 1$ for $1 \leq j \leq k$, and $z_j^- = -1$ for $j > k$. Then we have

$$D(\hat{\boldsymbol{\theta}}) \geq \frac{k}{n_+} + \sum_{i=1}^{k} \theta_i^+ + \sum_{j=1}^{k} \theta_j^- + \sum_{i \in \mathcal{P}} h_i^+(\theta_i^+) + \sum_{j \in \mathcal{N}} h_j^-(\theta_j^-).$$

Using $\theta^+ + \theta^- = \frac{-1}{n_+}$, we can compare:

$$D(\hat{\mathbf{\theta}}) - D(\mathbf{\theta}') \geq \sum_{i=1}^{k} \left[h_i^+(\theta_i^+) - h_i^+(\theta^+) + \theta_i^+ - \theta^+\right] + \sum_{j=1}^{k} \left[h_j^-(\theta_j^-) - h_j^-(\theta^-) + \theta_j^- - \theta^-\right] + \sum_{i=k+1}^{n_+} \left[h_i^+(\theta_i^+) - h_i^+(\theta^+)\right] + \sum_{j=k+1}^{n_-} \left[h_j^-(\theta_j^-) - h_j^-(\theta^-)\right]. \quad (29)$$

It is not hard to show all terms in the square bracket are non-negative. For $i \leq k$ and $j \leq k$, by definition of θ^+ and θ^-, we have $\theta_i^+ \geq \theta^+$ and $\theta_j^- \geq \theta^-$. Since the gradient of h_i^+ and h_j^- lies in $[-1,0]$, so by Rolle's mean value theorem we have

$$h_i^+(\theta_i^+) - h_i^+(\theta^+) + \theta_i^+ - \theta^+ \geq (-1)(\theta_i^+ - \theta^+) + \theta_i^+ - \theta^+ = 0,$$
$$h_j^-(\theta_j^-) - h_j^-(\theta^-) + \theta_j^- - \theta^- \geq (-1)(\theta_j^- - \theta^-) + \theta_j^- - \theta^- = 0.$$

For $i \geq k$ and $j \geq k$, by definition we have $\theta_i^+ \leq \theta^+$ and $\theta_j^- \leq \theta^-$, and so $h_i^+(\theta_i^+) \geq h_i^+(\theta^+)$ and $h_j^-(\theta_j^-) \geq h_j^-(\theta^-)$. By (29) we conclude $D(\hat{\mathbf{\theta}}) \geq D(\mathbf{\theta}')$.

Case 2: $k = n_+$. Then define $\theta^- = \theta_{n_+}^-$ and $\theta^+ = \frac{-1}{n_+} - \theta^-$. Since $k = n_+$ so $\theta^+ < \theta_{n_+}^+$. The rest of the proof for $D(\hat{\mathbf{\theta}}) \geq D(\mathbf{\theta}(\theta^+, \theta^-))$ is exactly the same as Case 1.

Case 3: $k = 0$. This means $\theta_i^+ + \theta_i^- + \frac{1}{n_+} \leq 0$ for all $1 \leq i \leq n_+$. Then define $\theta^+ = \theta_1^+$ and $\theta^- = \frac{-1}{n_+} - \theta^+$. Clearly, $\theta^- \geq \theta_1^-$. The rest of the proof for $D(\hat{\mathbf{\theta}}) \geq D(\mathbf{\theta}(\theta^+, \theta^-))$ is exactly the same as Case 1 with k set to 0.

Next we prove the second sentence of Theorem 7. We first show necessity. Denote $\mathbf{\theta}' := \mathbf{\theta}(\theta^+, \theta^-)$, and $\mathcal{Y} := \operatorname{argmax}_{\mathbf{z}} q(\mathbf{z}, \mathbf{\theta}')$. By the definition of PRBEP, any $\mathbf{z} \in \mathcal{Y}$ must satisfy

$$\sum_{i \in \mathcal{P}} \delta(z_i^+ = -1) = \sum_{j \in \mathcal{N}} \delta(z_j^- = 1). \quad (30)$$

Since $\mathbf{\theta}'$ is an optimizer of $D(\mathbf{\theta})$, so $\mathbf{0} \in \partial D(\mathbf{\theta}')$. Hence by (28), there must exist a distribution $\alpha(\mathbf{z})$ over $\mathbf{z} \in \mathcal{Y}$ such that

$$\nabla h_i^+(\theta^+) = -\sum_{\mathbf{z} \in \mathcal{Y}: z_i^+ = -1} \alpha(\mathbf{z}),\ \forall i \in \mathcal{P} \quad \text{and} \quad \nabla h_j^-(\theta^-) = -\sum_{\mathbf{z} \in \mathcal{Y}: z_j^- = 1} \alpha(\mathbf{z}),\ \forall j \in \mathcal{N}.$$

Therefore

$$\sum_{i \in \mathcal{P}} \nabla h_i^+(\theta^+) = -\sum_{i \in \mathcal{P}} \sum_{\mathbf{z} \in \mathcal{Y}: z_i^+ = -1} \alpha(\mathbf{z}) = -\sum_{\mathbf{z} \in \mathcal{Y}} \alpha(\mathbf{z}) \sum_{i \in \mathcal{P}} \delta(z_i^+ = -1)$$
$$\stackrel{\text{by (30)}}{=} -\sum_{\mathbf{z} \in \mathcal{Y}} \alpha(\mathbf{z}) \sum_{j \in \mathcal{N}} \delta(z_j^- = 1) = -\sum_{j \in \mathcal{N}} \sum_{\mathbf{z} \in \mathcal{Y}: z_j^- = 1} \alpha(\mathbf{z}) = \sum_{j \in \mathcal{N}} \nabla h_j^-(\theta^-).$$

Finally, we prove sufficiency. By the necessity, we know there is an optimal solution $\hat{\mathbf{\theta}} = \mathbf{\theta}(\theta^+, \theta^-)$ which satisfies $\theta^+ + \theta^- = \frac{-1}{n_+}$ and $\nabla f(\theta^+) = 0$. Now suppose there is another pair η^+ and η^- which also satisfy $\eta^+ + \eta^- = \frac{-1}{n_+}$ and $\nabla f(\eta^+) = 0$. Letting $\mathbf{\theta}' = \mathbf{\theta}(\eta^+, \eta^-)$, it suffices to show that $D(\mathbf{\theta}') = D(\hat{\mathbf{\theta}})$. Since $\theta^+ + \theta^- = \eta^+ + \eta^- = \frac{-1}{n_+}$, so

$$\max_{\mathbf{z}} q(\mathbf{z}, \hat{\mathbf{\theta}}) = \max_{\mathbf{z}} q(\mathbf{z}, \mathbf{\theta}') = 0. \quad (31)$$

Since f is convex and $\nabla f(\theta^+) = \nabla f(\eta^+) = 0$, so both θ^+ and η^+ minimize $f(\theta)$ and so $f(\theta^+) = f(\eta^+)$. In conjunction with (31), we conclude $D(\mathbf{\theta}') = D(\hat{\mathbf{\theta}})$. ■

4.1.2 AN $O(n\log n)$ COMPLEXITY ALGORITHM FOR SOLVING (26) UNDER PRBEP

We now design an $O(n\log n)$ complexity algorithm which, given $\{a_i\}$ from (27), finds an optimal solution $\hat{\boldsymbol{\theta}}$ exactly. By Theorem 7, it suffices to find a root of the gradient of f:

$$\nabla f(\theta) = g^+(\theta) - g^-\left(\frac{-1}{n_+} - \theta\right) \quad \text{where} \quad g^+(\theta) := \sum_{i\in\mathcal{P}} \nabla h_i^+(\theta) \ \text{ and } \ g^-(\theta) := \sum_{j\in\mathcal{N}} \nabla h_j^-(\theta).$$

Clearly, g^+ and g^- are piecewise linear. g^+ only has $2n_+$ kink points: $\{a_i^+, a_i^+ - \mu : i \in \mathcal{P}\}$ and we assume they are sorted decreasingly as $\tau_1^+ \geq \ldots \geq \tau_{2n_-}^+$. Similarly, g^- only has $2n_-$ kink points: $\{a_j^-, a_j^- - \mu : j \in \mathcal{N}\}$ and we assume they are sorted decreasingly as $\tau_1^- \geq \ldots \geq \tau_{2n_-}^-$. Further define $\tau_0^+ = \tau_0^- = \infty$ and $\tau_{2n_++1}^+ = \tau_{2n_-+1}^- = -\infty$.

Our main idea is to find a^+, b^+, a^-, b^- which satisfy i) g^+ is linear in $[a^+, b^+]$ and g^- is linear in $[a^-, b^-]$; and ii) there exists $\theta^+ \in [a^+, b^+]$ and $\theta^- \in [a^-, b^-]$ such that $\theta^+ + \theta^- = \frac{-1}{n_+}$ and $g^+(\theta^+) = g^-(\theta^-)$. Once this is done, we can find θ^+ and θ^- by solving a linear system

$$\theta^+ + \theta^- = \frac{-1}{n_+}\,,$$
$$g^+(a^+) + \frac{g^+(a^+) - g^+(b^+)}{a^+ - b^+}(\theta^+ - a^+) = g^-(a^-) + \frac{g^-(a^-) - g^-(b^-)}{a^- - b^-}(\theta^- - a^-). \tag{32}$$

The idea of finding a^+, b^+, a^-, b^- is to detect the sign switch of "some" function. Our framework can be divided into two stages:

- First, we find a sign switching interval $[\tau_i^+, \tau_{i-1}^+]$ such that $\nabla f(\tau_i^+) \leq 0$ and $\nabla f(\tau_{i-1}^+) \geq 0$. Formally, since $f(\tau_{2n_++1}^+) = \lim_{\theta\to-\infty} \nabla f(\theta) = -n_+$ and $f(\tau_0^+) = \lim_{\theta\to\infty} \nabla f(\theta) = n_-$, there must be a cross point $i \in \{1, \ldots, 1+2n_+\}$ such that $\nabla f(\tau_i^+) \leq 0$ and $\nabla f(\tau_{i-1}^+) \geq 0$. We set $a^+ = \tau_i^+$ and $b^+ = \tau_{i-1}^+$.

- Second, within $[a^+, b^+]$ where $c := \frac{-1}{n_+} - \tau_{i-1}^+$ and $d := \frac{-1}{n_+} - \tau_i^+$, we find a sign switching interval $[\tau_j^-, \tau_{j-1}^-]$ such that $\nabla f(\frac{-1}{n_+} - \tau_j^-) \geq 0$ and $\nabla f(\frac{-1}{n_+} - \tau_{j-1}^-) \leq 0$. Formally, suppose $\tau_p^- \geq \ldots \geq \tau_q^-$ are in $[c, d]$. Define a decreasing array by $\chi_0^- = d$, $\chi_1^- = \tau_q^-$, $\ldots$, $\chi_m^- = \tau_r^-$, $\chi_{m+1}^- = c$. Since $\nabla f(\frac{-1}{n_+} - \chi_0^-) = \nabla f(\tau_i^+) \leq 0$ and $\nabla f(\frac{-1}{n_+} - \chi_{m+1}^-) = \nabla f(\tau_{i-1}^+) \geq 0$, there must be a cross point $j \in \{1, \ldots, m+1\}$ such that $\nabla f(\frac{-1}{n_+} - \chi_j^-) \geq 0$ and $\nabla f(\frac{-1}{n_+} - \chi_{j-1}^-) \leq 0$. We set $a^- = \chi_j^-$ and $b^- = \chi_{j-1}^-$.

Then a^+, b^+, a^-, b^- are guaranteed to have the desired property. We formalize the idea in Algorithm 2. Since $\nabla f(\tau_0^+) = n_-$, it simply finds the smallest $i \geq 1$ such that $\nabla f(\tau_i^+) \leq 0$. Similarly, since $\nabla f(\frac{-1}{n_+} - \chi_0^-) \leq 0$, it just finds the smallest $j \geq 1$ such that $\nabla f(\frac{-1}{n_+} - \chi_j^-) \geq 0$.

Algorithm 2 requires an efficient way to evaluate g^+ and g^-. To this end, we design Algorithm 3 for computing g^+, while the case for g^- can be handled in sheer analogy. One key observation made from Algorithm 2 is that the location where g^+ is evaluated grows monotonically in the first and second stage. Therefore, our algorithm only needs to ensure that the *total* cost of evaluating g^+ in both stages is $O(n_+)$.

Our idea to achieve this is to maintain an anchor point $i \in \{1, \ldots, 2n_+\}$, together with the slope s and intercept/bias b of g^+ in the interval $[\tau_{i+1}^+, \tau_i^+]$. Whenever a query point θ is given, we move i

Algorithm 2: Find a^+, b^+, a^-, b^- to facilitate finding θ^+ and θ^- via solving (32).

Input: Arrays $\{\tau_i^+ : 0 \le i \le 2n_+ + 1\}$ and $\{\tau_j^- : 0 \le j \le 2n_- + 1\}$.
Output: a^+, b^+, a^-, b^- which satisfy the desired conditions.

for $i = 1, \ldots, 1 + 2n_+$ **do**
  $g_i^+ = g^+(\tau_i^+)$, $g_i^- = g^-\left(\frac{-1}{n_+} - \tau_i^+\right)$;
  if $g_i^+ - g_i^- \le 0$ **then** break;
Let $c = \frac{-1}{n_+} - \tau_{i-1}^+$, $d = \frac{-1}{n_+} - \tau_i^+$. Suppose $\tau_q^- \ge \ldots \ge \tau_r^-$ are in $[c, d]$;
Define a decreasing array $\chi_0^- = d$, $\chi_1^- = \tau_q^-, \ldots, \chi_m^- = \tau_r^-, \chi_{m+1}^- = c$ $(m = r - q + 1)$.
for $j = 1, \ldots, m + 1$ **do**
  $g_j^+ = g^+\left(\frac{-1}{n_+} - \chi_j^-\right)$, $g_j^- = g^-(\chi_j^-)$;
  if $g_j^+ - g_j^- \ge 0$ **then** break;
return $a^+ = \tau_i^+$, $b^+ = \tau_{i-1}^+$, $a^- = \chi_j^-$, $b^- = \chi_{j-1}^-$;

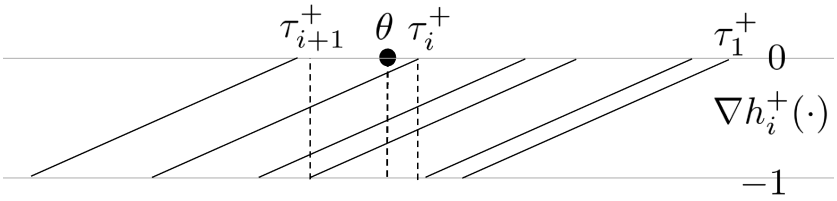


Figure 1: Example of evaluating g^+. At τ_i^+, $s = 3$ and $b = -2$. $g^+(\theta)$ can be computed by $-2 + 3(\theta - \tau_i^+)$. At τ_{i+1}^+, $s = 2$ and $b = -3$.

to the largest possible value such that $\theta \in (\tau_{i+1}^+, \tau_i^+]$ and compute $g^+(\theta)$ by $b + s(\theta - \tau_i^+)$. In other words, when $\theta > \tau_i^+$, we need to decrement i; and when $\theta \le \tau_{i+1}^+$ we increment i. An example is given in Figure 1.

The key operation here is how to update s and b when sliding i. We say i is a head if τ_i^+ corresponds to some a_k^+ (rather than $a_k^+ - \mu$). Otherwise we say i is a tail. We initialize by setting $i = 1$, $s = 1$ and $b = 0$. Then the update rule falls in four cases:

- Increment i and $i' = i + 1$ is a head: update by $s' = s + 1$, and $b' = b + s \cdot \frac{1}{\mu}(\tau_{i'}^+ - \tau_i^+)$.
- Increment i and $i' = i + 1$ is a tail: update by $s' = s - 1$ and $b' = b + s \cdot \frac{1}{\mu}(\tau_{i'}^+ - \tau_i^+)$.
- Decrement i to $i' = i - 1$, and i is a head: update by $s' = s - 1$ and $b' = b - s' \cdot \frac{1}{\mu}(\tau_i^+ - \tau_{i'}^+)$.
- Decrement i to $i' = i - 1$, and i is a tail: update by $s' = s + 1$ and $b' = b - s' \cdot \frac{1}{\mu}(\tau_i^+ - \tau_{i'}^+)$.

Since the query point grows monotonically, the anchor point i only needs to slide monotonically too. Hence the total computational cost in each stage is $O(n_+)$. It is also not hard to see that i can be initialized to $i = 2n_+$, with $s = 0$ and $b = -n_+$. Finally, suppose we store $\{a_i^+, a_i^+ - \mu\}$ by an array A with $A[2k-1] = a_k^+$ and $A[2k] = a_k^+ - \mu$, and the sorting is performed only in terms of the index of A. Then an index i of A is a head (corresponds to some a_k^+) if an only if i is odd.

Algorithm 3: Compute $g^+(\theta)$.

Input: A query point θ.
Output: $g^+(\theta)$.

Initialize: $i = 1$, $s = 1$, $b = 0$. Given $\{a_i\}$, do this only once at the first call of this function. Other initializations are also fine, for example, $i = 2n_+$, with $s = 0$ and $b = -n_+$.
while TRUE do
  if $\tau_i^+ \geq \theta$ **then**
    if $\tau_{i+1}^+ < \theta$ **then return** $b + s \cdot \frac{1}{\mu}(\theta - \tau_i^+)$;
    else
      $b = b + s \cdot \frac{1}{\mu}(\tau_{i+1}^+ - \tau_i^+)$, $i = i + 1$;
      if i is a head **then** $s = s + 1$; **else** $s = s - 1$;
  else
    if $i == 1$ **then return** 0;
    if i is a head **then** $s = s - 1$; **else** $s = s + 1$;
    $i = i - 1$, $b = b - s \cdot \frac{1}{\mu}(\tau_{i+1}^+ - \tau_i^+)$;

4.2 ROCArea Loss

For the ROCArea loss, given the optimal $\hat{\boldsymbol{\beta}}$ in (24) one can compute

$$\frac{\partial}{\partial \mathbf{w}} g_\mu^\star(A^\top \mathbf{w}) = \frac{1}{m} \sum_{i,j} \hat{\beta}_{ij}(\boldsymbol{x}_j - \boldsymbol{x}_i) = -\frac{1}{m}\left[\sum_{i \in \mathcal{P}} \boldsymbol{x}_i \underbrace{\left(\sum_{j \in \mathcal{N}} \hat{\beta}_{ij}\right)}_{:=\gamma_i} - \sum_{j \in \mathcal{N}} \boldsymbol{x}_j \underbrace{\left(\sum_{i \in \mathcal{P}} \hat{\beta}_{ij}\right)}_{:=\gamma_j}\right].$$

If we can efficiently compute all γ_i and γ_j, then the gradient can be computed in $O(np)$ time. Given $\hat{\beta}_{ij}$, a brute-force approach to compute γ_i and γ_j takes $O(m)$ time. We exploit the structure of the problem to reduce this cost to $O(n \log n)$, thus matching the complexity of the separation algorithm in Joachims (2005). Towards this end, we specialize (24) to ROCArea and write

$$g_\mu^\star(A^\top \mathbf{w}) = \max_{\boldsymbol{\beta}} \left(\frac{1}{m} \sum_{i,j} \beta_{ij} \mathbf{w}^\top (\boldsymbol{x}_j - \boldsymbol{x}_i) + \frac{1}{m} \sum_{i,j} \beta_{ij} - \frac{\mu}{2} \sum_{i,j} \beta_{ij}^2\right). \tag{33}$$

Since all β_{ij} are decoupled, their optimal value can be easily found:

$$\begin{aligned} \hat{\beta}_{ij} &= \text{median}\,(1, a_j - a_i, 0) \\ \text{where} \quad a_i &= \frac{1}{\mu m}\left(\mathbf{w}^\top \boldsymbol{x}_i - \frac{1}{2}\right), \text{ and } a_j = \frac{1}{\mu m}\left(\mathbf{w}^\top \boldsymbol{x}_j + \frac{1}{2}\right). \end{aligned} \tag{34}$$

Below we give a high level description of how γ_i for $i \in \mathcal{P}$ can be computed; the scheme for computing γ_j for $j \in \mathcal{N}$ is identical. We omit the details for brevity.

For a given i, suppose we can divide $\mathcal{N}$ into three sets $\mathcal{M}_i^+$, $\mathcal{M}_i$, and $\mathcal{M}_i^-$ such that

- $j \in \mathcal{M}_i^+ \implies 1 < a_j - a_i$, hence $\hat{\beta}_{ij} = 1$.

Algorithm 4: Compute γ_i, s_i, t_i for all $i \in \mathcal{P}$.

Input: Two arrays $\{a_i : i \in \mathcal{P}\}$ and $\{a_j : j \in \mathcal{N}\}$ sorted increasingly: $a_i \leq a_{i+1}$ and $a_j \leq a_{j+1}$.
Output: $\{\gamma_i, s_i, t_i : i \in \mathcal{P}\}$.

```
Initialize: s = t = 0, k = j = 0
for i in 1 to n_+ do
    while j < n_- AND a_{j+1} < a_i do
        j = j + 1; s = s - a_j; t = t - a_j^2
    while k < n_- AND a_{k+1} ≤ a_i + 1 do
        k = k + 1; s = s + a_k; t = t + a_k^2
    s_i = s, t_i = t, γ_i = (n_- - 1 - k) - (k - j)a_i + s
```

- $j \in \mathcal{M}_i \implies a_j - a_i \in [0,1]$, hence $\hat{\beta}_{ij} = a_j - a_i$.
- $j \in \mathcal{M}_i^- \implies a_j - a_i < 0$, hence $\hat{\beta}_{ij} = 0$.

Let $s_i = \sum_{j \in \mathcal{M}_i} a_j$. Then, clearly

$$\gamma_i = \sum_{j \in \mathcal{N}} \hat{\beta}_{ij} = |\mathcal{M}_i^+| + \sum_{j \in \mathcal{M}_i} a_j - |\mathcal{M}_i|\, a_i = |\mathcal{M}_i^+| + s_i - |\mathcal{M}_i|\, a_i.$$

Plugging the optimal $\hat{\beta}_{ij}$ into (33) and using (34), we can compute $g_\mu^\star(A^\top \mathbf{w})$ as

$$g_\mu^\star(A^\top \mathbf{w}) = \mu \sum_{ij} \hat{\beta}_{ij}(a_j - a_i) - \frac{\mu}{2} \sum_{ij} \hat{\beta}_{ij}^2 = \mu \left(\sum_{j \in \mathcal{N}} a_j \gamma_j - \sum_{i \in \mathcal{P}} a_i \gamma_i \right) - \frac{\mu}{2} \sum_{ij} \hat{\beta}_{ij}^2.$$

If we define $t_i := \sum_{j \in \mathcal{M}_i} a_j^2$, then $\sum_{ij} \hat{\beta}_{ij}^2$ can be efficiently computed via

$$\sum_j \hat{\beta}_{ij}^2 = \left|\mathcal{M}_i^+\right| + \sum_{j \in \mathcal{M}_i} (a_j - a_i)^2 = \left|\mathcal{M}_i^+\right| + \left|\mathcal{M}_i\right| a_i^2 - 2 a_i s_i + t_i, \quad \forall\, i \in \mathcal{P}.$$

In order to identify the sets $\mathcal{M}_i^+$, $\mathcal{M}_i$, $\mathcal{M}_i^-$, and compute s_i and t_i, we first sort both $\{a_i : i \in \mathcal{P}\}$ and $\{a_j : j \in \mathcal{N}\}$. We then walk down the sorted lists to identify for each i the first and last indices j such that $a_j - a_i \in [0,1]$. This is very similar to the algorithm used to merge two sorted lists, and takes $O(n_- + n_+) = O(n)$ time and space. The rest of the operations for computing γ_i can be performed in $O(1)$ time with some straightforward book-keeping. The whole algorithm is presented in Algorithm 4 and its overall complexity is dominated by the complexity of sorting the two lists, which is $O(n \log n)$.

5. Empirical Evaluation

We used 24 publicly available data sets and focused our study on two aspects: the reduction in objective value as a function of CPU time, and generalization performance. Since the objective functions we are minimizing are strongly convex, all optimizers will converge the same solution (within numerical precision) and produce the same generalization performance eventually.

Therefore, what we are specifically interested in is the rate at which the objective function and generalization performance decreases. We will refer to our algorithm as SMS, for **S**moothing for **M**ultivariate **S**cores. As our comparator we use BMRM. We downloaded BMRM from `http://users.rsise.anu.edu.au/~chteo/BMRM.html`, and used the default settings in all our experiments.

5.1 Data Sets

Table 1 summarizes the data sets used in our experiments. `adult9`, `astro-ph`, `news20`, `real-sim`, `reuters-c11`, `reuters-ccat` are from the same source as in Hsieh et al. (2008). `aut-avn` is from Andrew McCallum's home page,[6] `covertype` is from the UCI repository (Frank and Asuncion, 2010), `worm` is from Franc and Sonnenburg (2008), `kdd99` is from KDD Cup 1999,[7] while `web8`, `webspam-u`, `webspam-t`,[8] as well as the `kdda` and `kddb`[9] are from the LibSVM binary data collection.[10] The `alpha`, `beta`, `delta`, `dna`, `epsilon`, `fd`, `gamma`, `ocr`, and `zeta` data sets were all obtained from the Pascal Large Scale Learning Workshop website (Sonnenburg et al., 2008). Since the features of `dna` are suboptimal compared with the string kernels used by Sonnenburg and Franc (2010), we downloaded their original DNA sequence (`dna_string`).[11] In particular, we used a weighted degree kernel and two weighted spectrum kernels (one at position 1-59 and one at 62-141, corresponding to the left and right of the splice site respectively), all with degree 8. Following Sonnenburg and Franc (2010), we used the dense explicit feature representations of the kernels, which amounted to 12,670,100 features.

For `dna_string` and the data sets which were also used by Teo et al. (2010) (indicated by an asterisk in Table 1), we used the training test split provided by them. For the remaining data sets we used 80% of the labeled data for training and the remaining 20% for testing. In all cases, we added a constant feature as a bias.

5.2 Optimization Algorithms

Optimizing the smooth objective function $J_\mu(\mathbf{w})$ using the optimization scheme described in Nesterov (2005) requires estimating the Lipschitz constant of the gradient of the $g^\star_\mu(A^\top \mathbf{w})$. Although it can be automatically tuned by, for example, Beck and Teboulle (2009), extra cost is incurred which slows down the optimization empirically. Therefore, we chose to optimize our smooth objective function using L-BFGS, a widely used quasi-Newton solver (Nocedal and Wright, 1999). We implemented our smoothed loss using PETSc[12] and TAO,[13] which allow the *efficient* use of large-scale parallel linear algebra. We used the Limited Memory Variable Metric (`lmvm`) variant of

6. The data set can be found at `http://www.cs.umass.edu/~mccallum/data/sraa.tar.gz`.
7. The data set can be found at `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.
8. `webspam-u` is the webspam-unigram and `webspam-t` is the webspam-trigram data set. Original data set can be found at `http://www.cc.gatech.edu/projects/doi/WebbSpamCorpus.html`.
9. These data sets were derived from KDD CUP 2010. `kdda` is the first problem algebra_2008_2009 and `kddb` is the second problem bridge_to_algebra_2008_2009.
10. The data set can be found at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`.
11. The data set can be found at `http://sonnenburgs.de/soeren/projects/coffin/splice_data.tar.xz`.
12. The software can be found at `http://www.mcs.anl.gov/petsc/petsc-2/index.html`. We compiled an optimized version of PETSc (`--with-debugging=0`) and enabled 64-bit index to run for large data sets such as `dna` and `ocr`.
13. TAO can be found at `http://www.mcs.anl.gov/research/projects/tao`.

data set	n	p	$s(\%)$	$n_+ : n_-$	data set	n	p	$s(\%)$	$n_+ : n_-$
adult9*	48,842	123	11.3	0.32	alpha	500,000	500	100	1.00
astro-ph*	94,856	99,757	0.08	0.31	aut-avn*	71,066	20,707	0.25	1.84
beta	500,000	500	100	1.00	covertype*	581,012	6.27 M	22.22	0.57
delta	500,000	500	100	1.00	dna	50.00 M	800	25	3e−3
dna_string	50.00 M	12.7 M	*n.a.*	3e−3	epsilon	500,000	2000	100	1.00
fd	625,880	900	100	0.09	gamma	500,000	500	100	1.00
kdd99*	5.21 M	127	12.86	4.04	kdda	8.92 M	20.22 M	2e−4	5.80
kddb	20.01 M	29.89 M	1e−4	6.18	news20*	19,954	7.26 M	0.033	1.00
ocr	3.50 M	1156	100	0.96	real-sim*	72,201	2.97 M	0.25	0.44
reuters-c11*	804,414	1.76 M	0.16	0.03	reuters-ccat*	804,414	1.76 M	0.16	0.90
web8*	59,245	300	4.24	0.03	webspam-t	350,000	16.61 M	0.022	1.54
webspam-u	350,000	254	33.8	1.54	worm*	1.03 M	804	25	0.06
zeta	500,000	800.4 M	100	1.00					

Table 1: Summary of the data sets used in our experiments. n is the total number of examples, p is the number of features, s is the feature density (% of features that are non-zero), and $n_+ : n_-$ is the ratio of the number of positive vs negative examples. M denotes a million. A data set is marked with an asterisk if it is also used by Teo et al. (2010).

L-BFGS which is implemented in TAO. Open-source code, as well as all the scripts used to run our experiments are available for download from Zhang et al. (2012).

5.3 Implementation And Hardware

Both BMRM and SMS are implemented in C++. Since dna and ocr require more memory than was available on a single machine, we ran both BMRM and SMS with 16 cores spread across 16 machines for these two data sets. As dna_string employs a large number of dense features, it would cost a prohibitively large amount of memory. So following Sonnenburg and Franc (2010), we (repeatedly) computed the explicit features whenever it is multiplied with the weight vector $\boldsymbol{w}$. This entails demanding cost in computation, and therefore we used 32 cores. The other data sets were trained with a single core. All experiments were conducted on the Rossmann computing cluster at Purdue University,[14] where each node has two 2.1 GHz 12-core AMD 6172 processors with 48 GB physical memory per node.

5.4 Experimental Setup

We used $\lambda \in \{10^{-2}, 10^{-4}, 10^{-6}\}$ to test the performance of BMRM and SMS under a reasonably wide range of λ. In line with Chapelle (2007), we observed that solutions with higher accuracy do not improve the generalization performance, and setting $\varepsilon = 0.001$ was often sufficient. Accordingly, we could set $\mu = \varepsilon / D$ as suggested by the uniform deviation bound (13). However, this estimate is very conservative because in (13) we used D as an upper bound on the prox-function, while in practice the quality of the approximation depends on the value of the prox-function *around the optimum*. On the other hand, a larger value of μ proffers more strong convexity in g_μ, which makes

14. The cluster information is at http://www.rcac.purdue.edu/userinfo/resources/rossmann.

data set	PRBEP		ROCArea		data set	PRBEP		ROCArea	
	BMRM	SMS	BMRM	SMS		BMRM	SMS	BMRM	SMS
adult9	43.32	0.40	1.13	0.19	alpha	3595	183.6	802.3	135.4
astro-ph	119.0	48.8	11.8	4.02	aut-avn	2.22	1.77	0.92	0.61
beta	1710	24.4	644.0	46.5	covertype	36.24	7.33	42.73	5.71
delta	*n.a.*	36.03	*n.a.*	44.20	dna	*n.a.*	*n.a.*	12140	561.5
dna_string	*n.a.*	*n.a.*	>30000	27514	epsilon	1716.1	804.5	1182.4	718.4
fd	943.2	154.0	1329.8	327.5	gamma	14612	26.5	8902.0	47.5
kdd99	63.5	152.1	204.9	31.0	kdda	*n.a.*	19204	*n.a.*	6108
kddb	*n.a.*	19363	*n.a.*	10592	news20	6921.2	429.5	1.51	8.8
ocr	893.1	105.9	921.6	98.0	real-sim	1.19	3.52	8.50	2.16
reuters-c11	45.5	46.1	28.7	10.4	reuters-ccat	328.2	162.8	145.0	31.9
web8	4.35	1.26	10.4	0.59	webspam-t	17271	5681.2	5751.9	1805.4
webspam-u	57.8	29.5	58.8	8.47	worm	2058.6	30.4	1901.4	20.2
zeta	679.1	293.4	579.8	386.7					

Table 2: Summary of the wall-clock time taken by BMRM and SMS to find a solution which is within 1% of the minimal regularized risk. All numbers are in seconds, and are obtained from the λ which yields the optimal test performance. There are a few *n.a.* which are explained in Section 5.5.

g^*_μ smoother and hence J_μ easier to optimize. Empirically, we observed that setting $\mu = 10 \cdot \varepsilon / D$ allows SMS to find an ε accurate solution in all cases.

5.5 Results

In Figures 2 to Figure 26 we plot the evolution of the non-smooth objective $J(\mathbf{w})$ (1) as a function of CPU time as well as the generalization performance on the test set for both BMRM and SMS. For a fair comparison the time taken to load the data is excluded for both algorithms. Note that the x-axis on the plots scales logarithmically. The figures also indicate the generalization performance upon convergence, and the best performance among all values of λ is highlighted in boldface.

Across a variety of data sets, for both PRBEP and ROCArea problems, SMS reduces the non-smooth objective $J(\mathbf{w})$ significantly faster than BMRM for all values of λ. For large values of λ (e.g., $\lambda = 10^{-2}$) the difference in performance is not dramatic, but the test performance of both methods is always inferior. This is because most of the data sets we use in our experiments contain a large number of training examples, and hence require very mild regularization. For instance, the optimal generalization performance is obtained by $\lambda = 10^{-6}$ in 40 out of the 48 experiments, by $\lambda = 10^{-4}$ in 14 experiments, and $\lambda = 10^{-2}$ in only 2 experiments (there are 8 ties). For low values of λ we find that BMRM which sports a $O\left(\frac{1}{\lambda\varepsilon}\right)$ rate of convergence slows down significantly. In contrast, SMS is not only able to gracefully handle small values of λ but actually converges significantly faster. To summarize the results, we present in Table 2 the wall-clock time that BMRM and SMS take to minimize the regularized risk to 1% relative accuracy, that is, find $\boldsymbol{w}^*$ such that $J(\boldsymbol{w}^*) \leq 1.01 \cdot \min_{\boldsymbol{w}} J(\boldsymbol{w})$. In many cases, SMS is 10 to 100 times faster than BMRM in achieving the same relative accuracy. Also see the detailed discussion of the results below.

We also studied the evolution of the PRBEP and ROCArea performance on the test data. It is clear that *in general* a lower value in $J(\boldsymbol{w})$ results in better test performance, which confirms the effectiveness of the SVM model. This correlation also translates the faster reduction of the objective value into the faster improvement of the generalization performance. On most data sets, the intermediate models output by SMS achieve comparable (or better) PRBEP and ROCArea scores in time orders of magnitude faster than those generated by BMRM. These results can be explained as follows: Since BMRM is a CPM, only the duality gap is guaranteed to decrease monotonically. This does not translate to reduction in the primal objective value. Consequently, we observe that sometimes the primal objective and generalization performance of BMRM seemingly gets "stuck" for many iterations. In contrast, our smoothing scheme uses L-BFGS which enforces a descent condition, and hence we observe a monotonic decrease in objective value at every iteration.[15]

5.5.1 DETAILED DISCUSSION OF THE RESULTS

On the `adult9` data set (Figure 2), the value of λ that yields the optimal test PRBEP is 10^{-6} (panels (c) and (f)). 100 seconds are needed by BMRM to complete training (i.e., minimize the regularized risk $J(\boldsymbol{w})$ to the desired accuracy and achieve stably optimal test PRBEP), while SMS needs only 0.5 seconds. For ROCArea, the optimal λ is 10^{-4} (panels (h) and (k)), in which case it takes SMS just 0.3 seconds for training, compared with the 2 seconds required by BMRM.

On the `covertype` data set (Figure 7), the optimal λ is 10^{-6} for both PRBEP and ROCArea. In PRBEP-problem (panels (c) and (f)), SMS is able to find the optimal regularized risk in 15 seconds, while BMRM takes about 55 seconds. The optimal test PRBEP is stably achieved by SMS in 10 seconds, while that costs BMRM about 7 times longer (80 seconds). The superiority on ROCArea is even more evident (panels (i) and (l)). SMS takes only 10 seconds to complete training for ROCArea, while BMRM requires 60 seconds. In addition, between 1 second and 10 seconds, BMRM seems to be stuck by a plateau where it is collecting cutting planes to construct a refined model before being able to reduce the regularized risk further.

On the `delta` data set (Figure 8), the optimal λ is again 10^{-6} for both PRBEP and ROCArea. In both cases, BMRM fails to converge to the optimal solution within 9 hours' runtime and is terminated (panels (c), (f), (i), (l), and the "*n.a.*" for `delta` in Table 2). In contrast, SMS optimizes the objective and attains the optimal test PRBEP and ROCArea in only 40 seconds. If we also pay attention to the sub-optimal values of λ, that is, 10^{-2} and 10^{-4}, it is clear that SMS again optimizes the regularized risk and test performance much faster. In fact, the time required by SMS to output the optimal test performance is about only 1% of that of BMRM (panels (d), (e), (j), (k)). Similar phenomena are observed on the `gamma` data set (Figure 13).

`ocr` is another data set where SMS demonstrates a clear edge over BMRM (Figure 18). The optimal test PRBEP and ROCArea are both attained at $\lambda = 10^{-6}$. Training for PRBEP costs BMRM 1000 seconds, while SMS takes only 10% as much time (panels (c) and (f)). As for ROCArea, the training time required by BMRM is 10 times of that of SMS. Similar striking advantage of SMS is also manifested on the `fd` data set (Figure 12). As one can clearly read from panels (c), (f), (i), and (l), SMS completes training for PRBEP in only 150 seconds, 15% of the time needed by BMRM. On ROCArea, training costs SMS 200 seconds, as opposed to 800 seconds for BMRM.

15. L-BFGS enforces monotonic decrease only on the smoothed objective $J_\mu(\boldsymbol{w})$, while what we are plotting is $J(\boldsymbol{w})$. However, empirically we observe that $J(\boldsymbol{w})$ is always monotonically decreasing.

On the two reuters data sets, the advantage of SMS is also manifested. On reuters-c11 (Figure 20), the model achieves about 50% test PRBEP (panel (f)). Since the data set is imbalanced with the ratio of the number of positive versus negative examples being 0.03, this performance is probably not vacuous. SMS and BMRM perform similarly in minimizing the regularized risk (panel (c)), while the test PRBEP fluctuates too much making it hard to tell which method is better (panel (f)). Regarding ROCArea, SMS is clearly faster in reducing the regularized risk (panel (i)), but in terms of test ROCArea it is only slightly faster than BMRM (panel (l)). The advantage of SMS is much clearer on the reuters-ccat data set (Figure 21), whose class ratio is 0.9. With $\lambda = 10^{-6}$ which yields the optimal PRBEP, SMS completes training in about 110 seconds, while BMRM takes 400 seconds (panel (c) and (f)). For ROCArea, the optimal λ is again 10^{-6} and 30 seconds are needed by SMS to complete training, compared with the 100 seconds required by BMRM (panels (i) and (l)).

Let us look at another pair of similar data sets: webspam-u and webspam-t. On webspam-u (Figure 24), SMS demonstrates clear superiority over BMRM. The optimal test PRBEP and ROCArea are both achieved at $\lambda = 10^{-6}$, which outperforms the result of $\lambda = 10^{-4}$ by 0.72% and 0.73% respectively (panels (f) and (l)). As far as ROCArea is concerned, SMS completed training in 9 seconds, while BMRM takes 30 seconds (panels (i) and (l)). In the PRBEP-problem (panels (c) and (f)), training takes 250 seconds for SMS, while the time cost for BMRM is doubled (500 seconds). Very similar results are observed on webspam-t (Figure 23) which differs from webspam-u by using a trigram class of features. On PRBEP, the regularized risk is minimized by SMS in 6,500 seconds (panels (c)), while BMRM requires approximately three times more CPU time (20,000 seconds). The optimal test PRBEP is achieved by SMS in 2,000 seconds (panel (f)), while BMRM takes double as much time (4,000 seconds). Similarly on ROCArea (panels (i) and (l)), SMS is three times more efficient than BMRM, taking 2,000 seconds to minimize the regularized risk (versus 7,000 seconds by BMRM), and 7,000 seconds to optimize the test ROCArea (versus 25,000 seconds by BMRM).

In the PRBEP-problem for kdda and kddb, BMRM is stuck half way because its inner solver for quadratic programming does not converge even after a large number of iterations, and is terminated after running for 9 hours. Hence we mark "*n.a.*" in Table 2. We tried both the QR-LOQO and the Dai_Fletcher solvers provided by BMRM, but they both got stuck. However, SMS had no trouble solving the PRBEP-problem efficiently. Turning our attention to the ROCArea-problems, the optimal λ is 10^{-6} for both kdda and kddb. Here BMRM optimizes the objective very slowly and cannot get close to the optimal solution within 9 hours, hence terminated. SMS again solves the ROCArea-problems efficiently. Note when λ is large (10^{-4} and 10^{-2}), BMRM does manage to find the optimal solution for the ROCArea-problems within 9 hours, but SMS completes training in less than a quarter of the time taken by BMRM.

As is to be expected in such extensive empirical evaluation, there are some anomalies. On the dna data set (Figure 9), the ROCArea is learned very well (panels (j) to (l)), whereas the test PRBEP is pretty poor (panels (d) to (f)). We have enforced the accuracy of regularized risk minimization to very high: 10^{-5}, but in panels (a) to (c), the objective value at convergence is still just slightly below 1, a value which is trivially attained by $\boldsymbol{w} = \mathbf{0}$. Hence we mark "*n.a.*" in Table 2. We tried with smaller λ (e.g., 10^{-8} and 10^{-10}) and got similar test PRBEP. It is noteworthy that dna is the most skewed data set used in our experiment, with 333 times more negatives examples than positive examples. BMRM also struggles on this data set, which leads us to believe that some other learning models may be needed. Similar behavior is also observed on the dna_string data set, which attains

significantly higher test PRBEP and ROCArea than dna does, confirming the superiority of string kernels.

On the beta data set (Figure 6), the optimal PRBEP and ROCArea are around 50% (panels (f) and (l)). Since this data set contains the same number of positive and negative examples, a random labeling of the test data will produce 50% PRBEP and ROCArea in expectation. So for this data set, other modeling methods may be needed as well.

On kdd99 (Figure 14), the objective value is not well correlated with the test performance. For example, although SMS is clearly faster than BMRM in reducing the regularized risk for ROCArea (panels (g) to (i)), the test ROCArea is improved at similar rates by the two algorithms. In fact, the test ROCArea of SMS in panel (l) is even worse. Likewise, SMS and BMRM perform similarly in reducing the regularized risk for PRBEP as shown in panels (a) to (c), but irregular bumps in the test PRBEP are observed on BMRM (panels (d) to (f)). Notice that at convergence, both algorithms still match closely in the test performance.

5.5.2 THE INFLUENCE OF μ

Figures 27 to 29 show the performance of SMS under different values of μ with λ fixed to 10^{-6}. The complete results on all the 24 data sets are available at Zhang et al. (2012).

We tried $\mu = c_\mu \cdot \varepsilon / D$ where $c_\mu \in \{1000, 100, 10\}$. It is clear that $c_\mu = 10$ can always find an ε accurate solution in practice. This guarantee fails to hold when c_μ is too large (e.g., 1000), although in this case SMS sometimes converges faster and may also generalize well on test data. $c_\mu = 100$ often performs very similarly to $c_\mu = 10$.

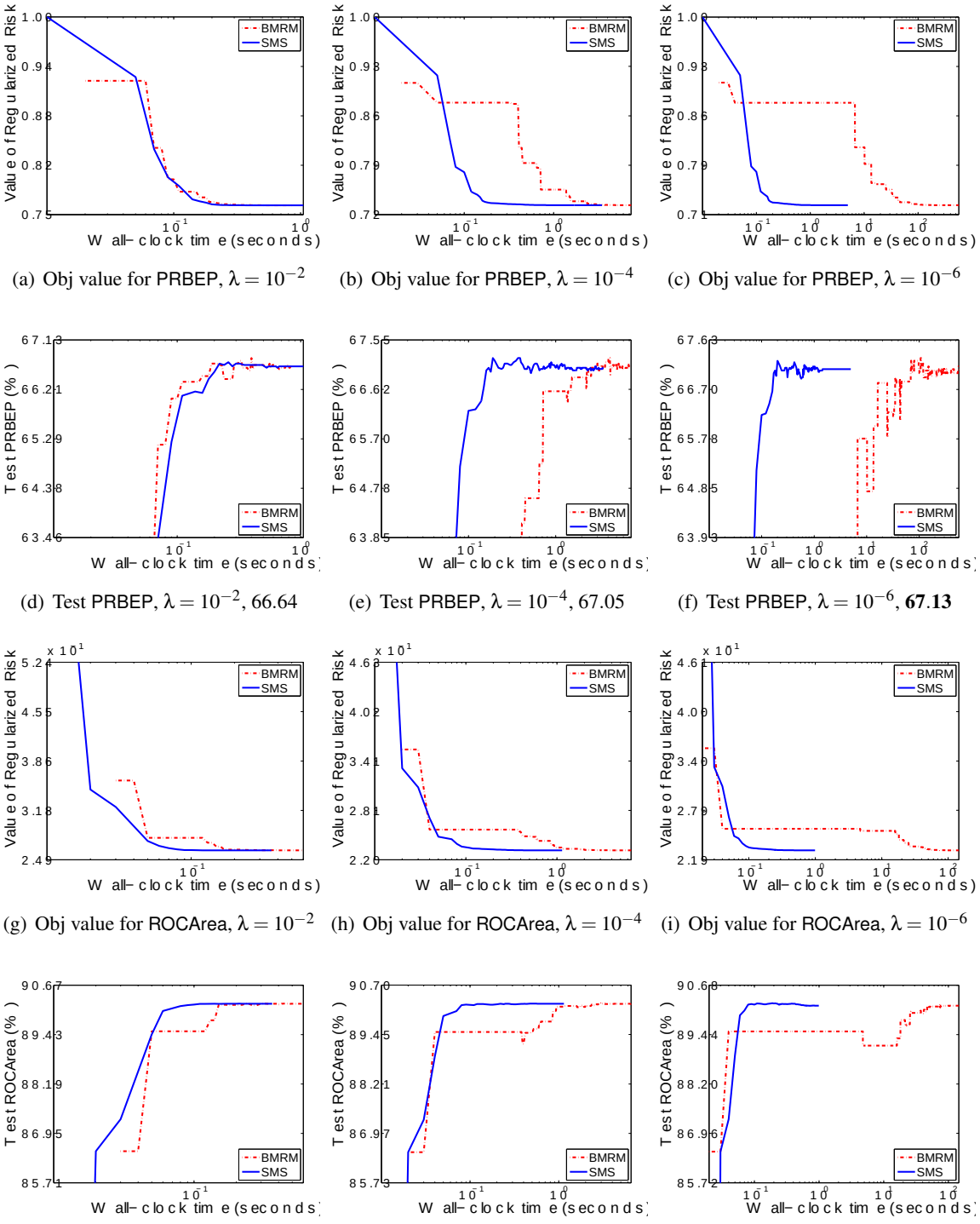
5.5.3 COMPARISON WITH ACCELERATED GRADIENT METHOD

Motivated by the theoretical derivation of the rates of convergence, we tested the performance of Nesterov's accelerated gradient method (AGM) on our smooth objective. In particular, we used the scheme in Nesterov (2007) which adaptively estimates the Lipschitz constant of the gradient. From the results in Figure 30 to 32 where $\lambda = 10^{-6}$ and $c_\mu = 10$, it is clear that AGM converges much slower than L-BFGS. This is not surprising because L-BFGS, which is a second order method, can estimate and use the curvature of the unconstrained and smooth objective. The full set of results on all data sets are available at Zhang et al. (2012).

6. Conclusion And Discussion

The non-smoothness of the loss function is an important consideration for algorithms which employ the kernel trick (Schölkopf and Smola, 2002). This is because such algorithms typically operate in the dual, and the non-smooth losses lead to sparse dual solutions. In many applications such as natural language processing, the kernel trick is not needed because the input data is sufficiently high dimensional. However, now we need to tackle a non-smooth optimization problem in the primal. In this paper we proposed efficient smoothing techniques to approximate the non-smooth function. When combined with a smooth optimization algorithm, our technique outperforms state-of-the-art non-smooth optimization algorithms for multivariate performance scores not only in terms of CPU time but also in terms of generalization performance.

It is also worthwhile emphasizing here that optimization is a means to an end in Machine Learning, and smoothing is not the right approach for every non-smooth problem. For example, although



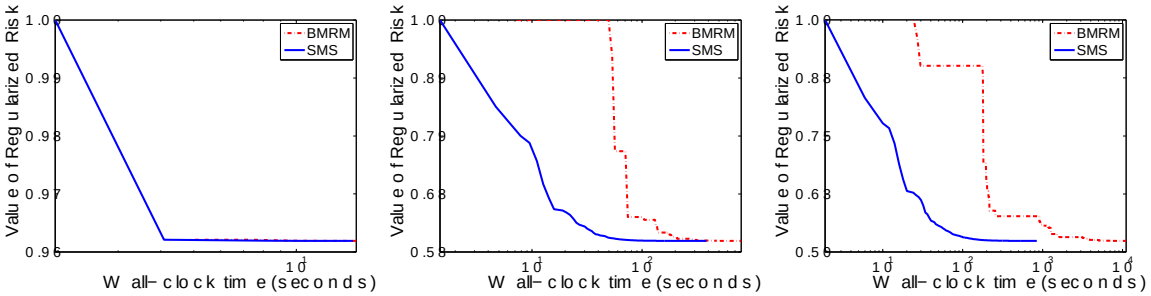
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$

(d) Test PRBEP, $\lambda = 10^{-2}$, 66.64 (e) Test PRBEP, $\lambda = 10^{-4}$, 67.05 (f) Test PRBEP, $\lambda = 10^{-6}$, **67.13**

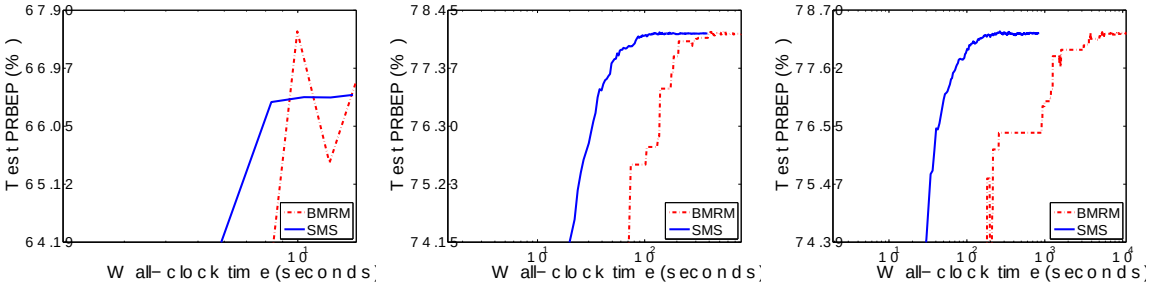
(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

(j) Test ROCArea, $\lambda = 10^{-2}$, 90.21 (k) Test ROCArea, $\lambda = 10^{-4}$, **90.23** (l) Test ROCArea, $\lambda = 10^{-6}$, 90.17

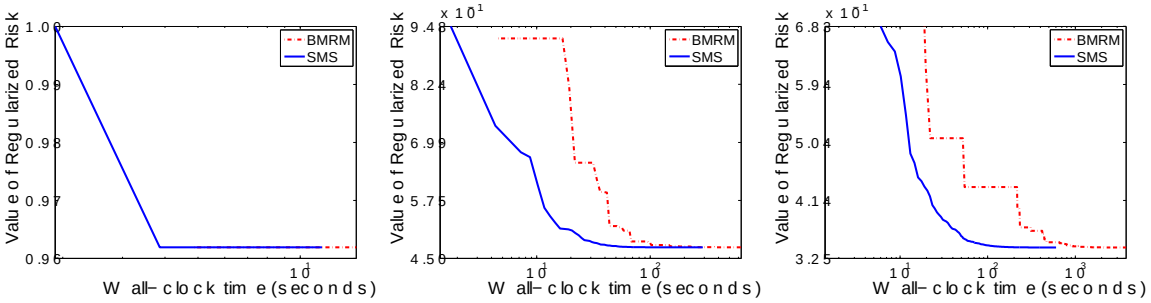
Figure 2: Results for `adult9`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



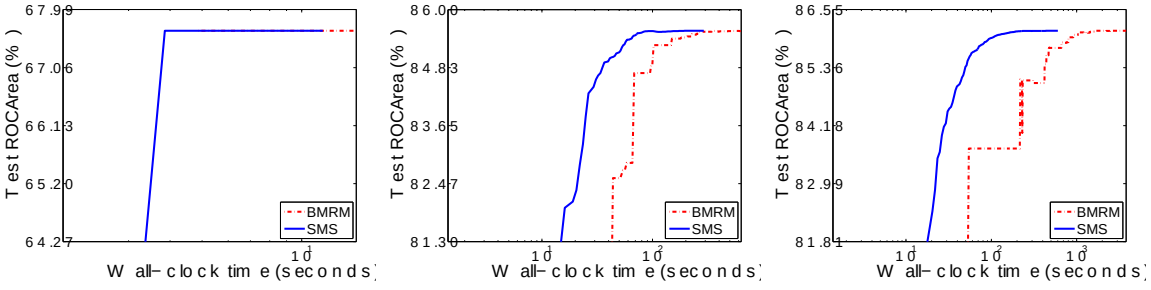
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 66.55 (e) Test PRBEP, $\lambda = 10^{-4}$, 78.00 (f) Test PRBEP, $\lambda = 10^{-6}$, **78.27**

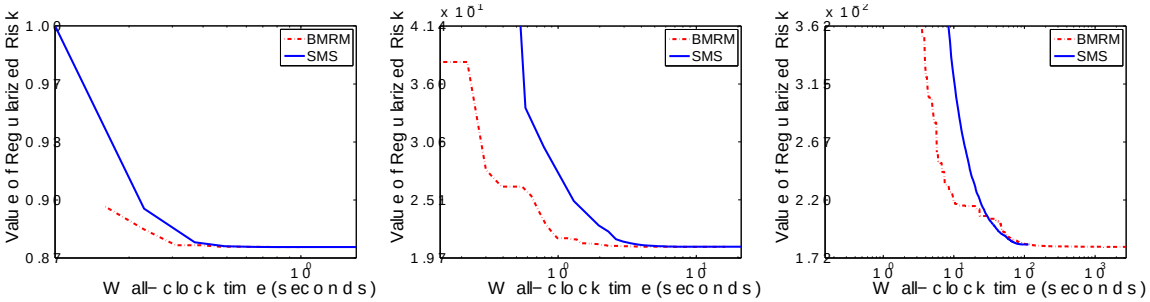


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

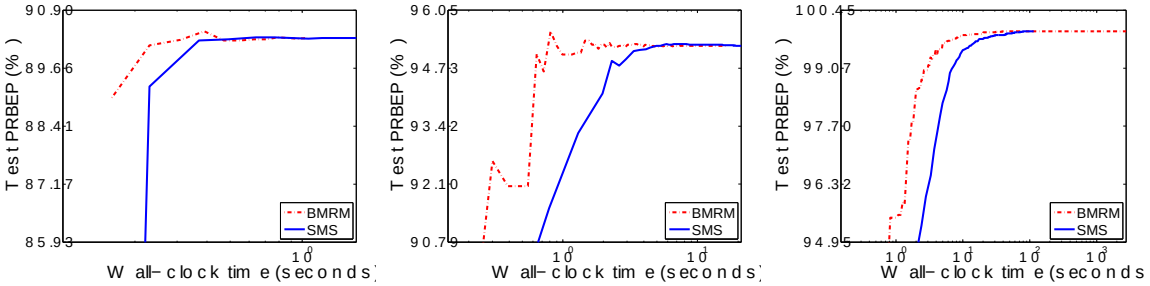


(j) Test ROCArea, $\lambda = 10^{-2}$, 61.84 (k) Test ROCArea, $\lambda = 10^{-4}$, 85.57 (l) Test ROCArea, $\lambda = 10^{-6}$, **86.12**

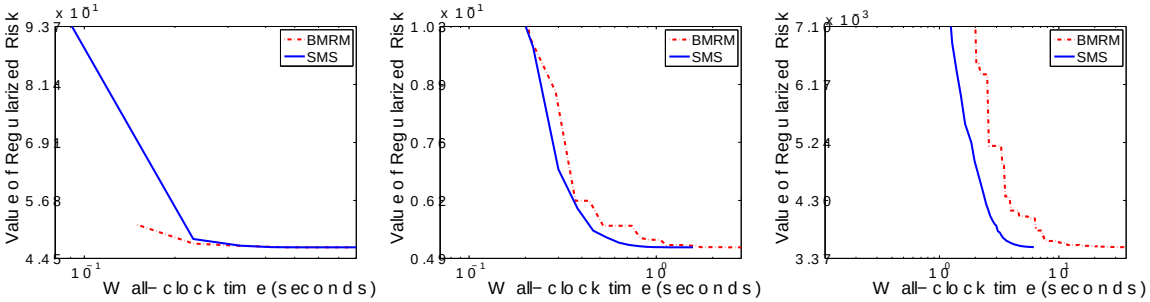
Figure 3: Results for `alpha`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



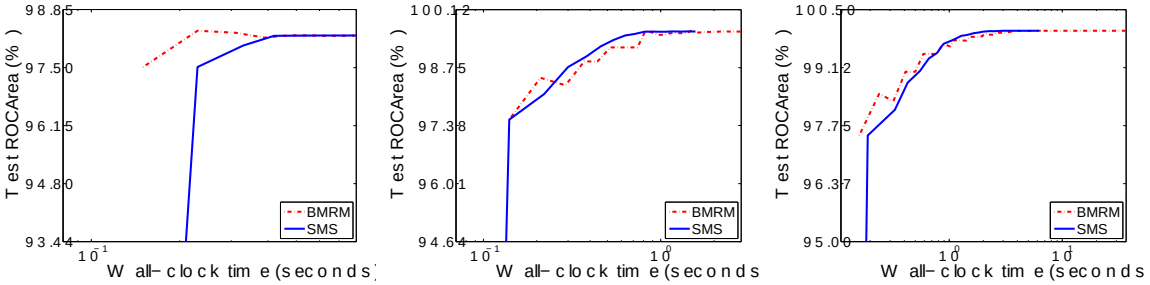
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 90.30 (e) Test PRBEP, $\lambda = 10^{-4}$, 95.24 (f) Test PRBEP, $\lambda = 10^{-6}$, **99.95**

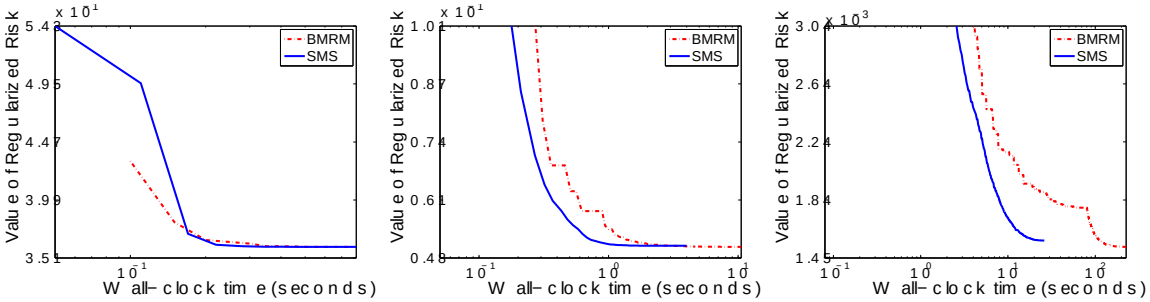


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

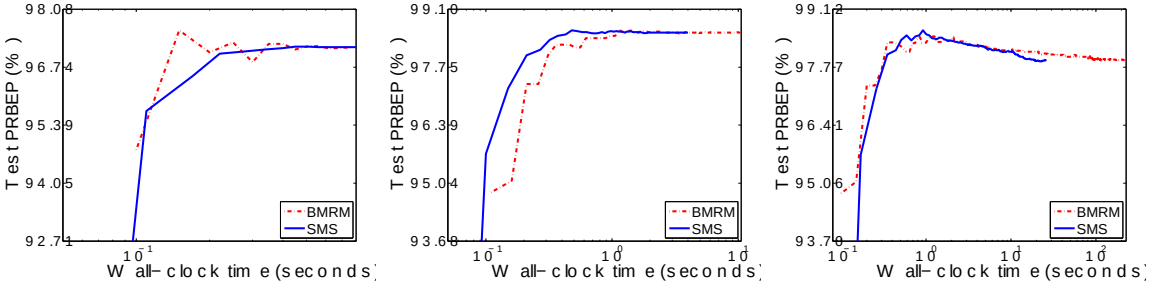


(j) Test ROCArea, $\lambda = 10^{-2}$, 98.25 (k) Test ROCArea, $\lambda = 10^{-4}$, 99.60 (l) Test ROCArea, $\lambda = 10^{-6}$, **100.00**

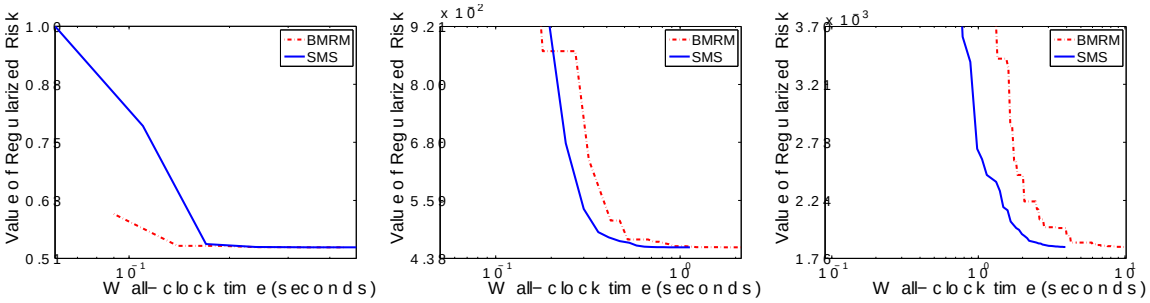
Figure 4: Results for `astro-ph`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



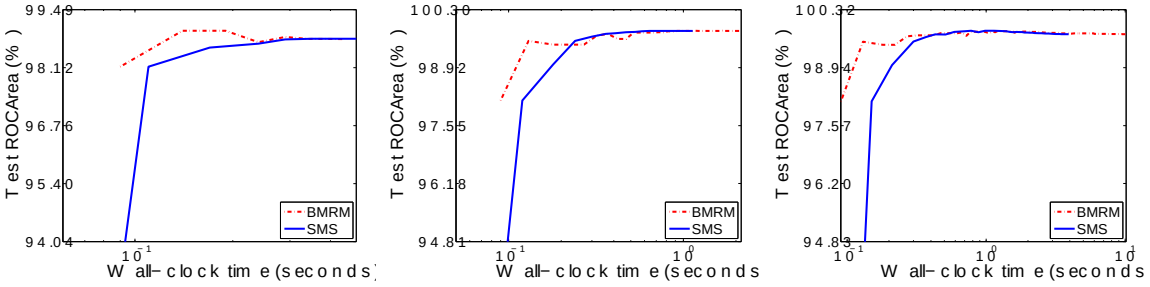
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 97.20 (e) Test PRBEP, $\lambda = 10^{-4}$, **98.56** (f) Test PRBEP, $\lambda = 10^{-6}$, 97.93

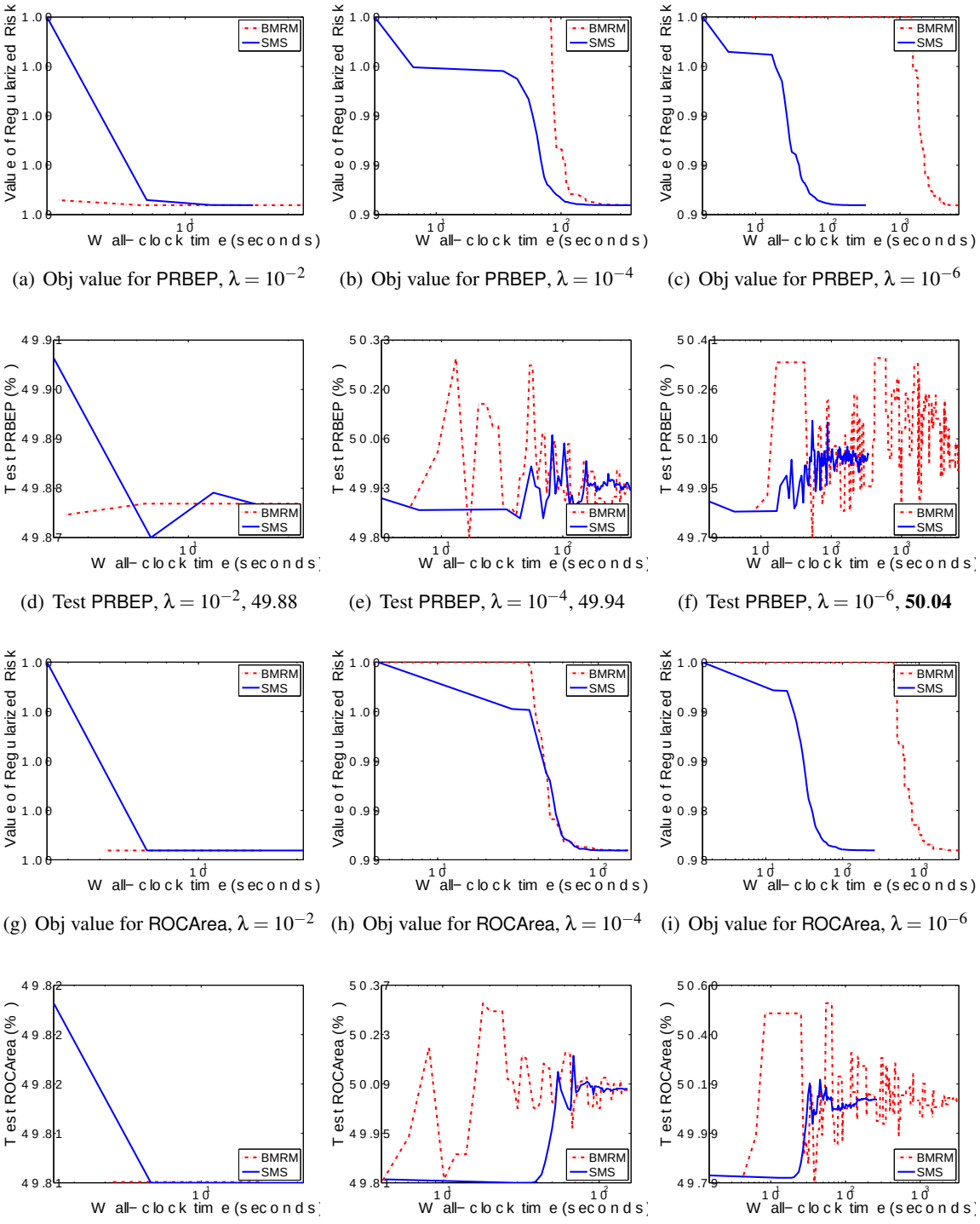


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$



(j) Test ROCArea, $\lambda = 10^{-2}$, 98.80 (k) Test ROCArea, $\lambda = 10^{-4}$, **99.79** (l) Test ROCArea, $\lambda = 10^{-6}$, 99.73

Figure 5: Results for `aut-avn`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



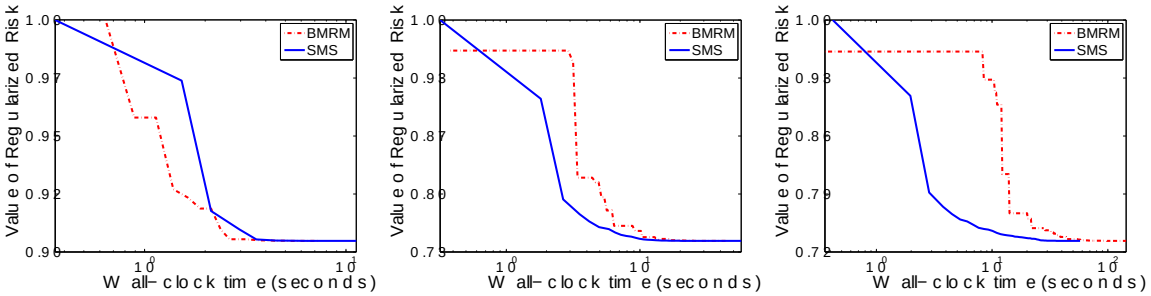
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$

(d) Test PRBEP, $\lambda = 10^{-2}$, 49.88 (e) Test PRBEP, $\lambda = 10^{-4}$, 49.94 (f) Test PRBEP, $\lambda = 10^{-6}$, **50.04**

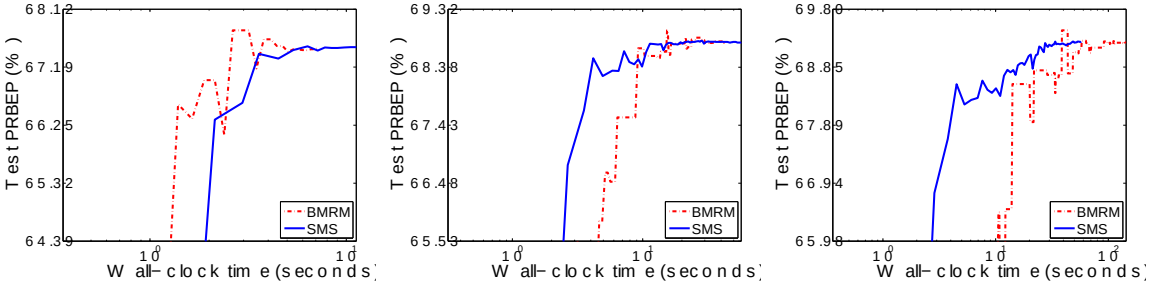
(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

(j) Test ROCArea, $\lambda = 10^{-2}$, 49.81 (k) Test ROCArea, $\lambda = 10^{-4}$, 50.07 (l) Test ROCArea, $\lambda = 10^{-6}$, **50.13**

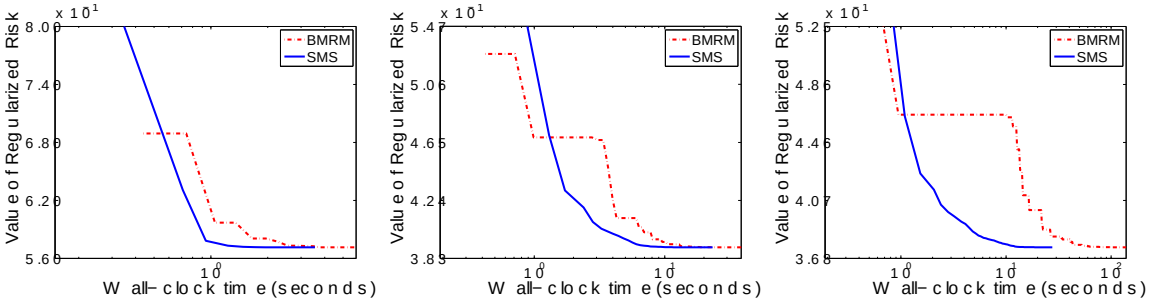
Figure 6: Results for `beta`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



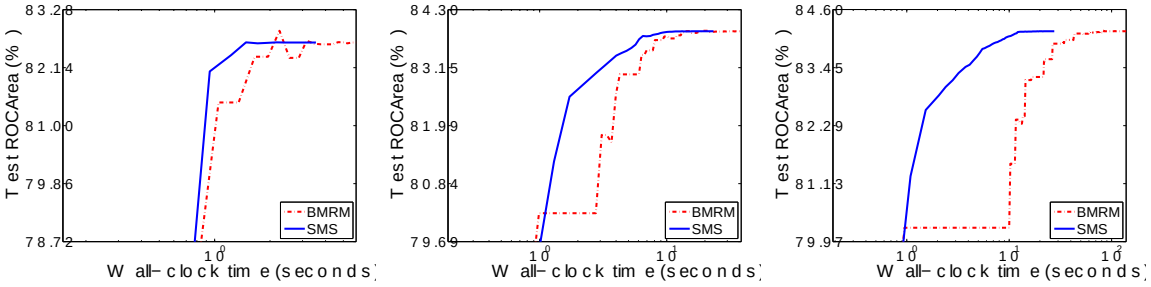
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 67.51 (e) Test PRBEP, $\lambda = 10^{-4}$, 68.78 (f) Test PRBEP, $\lambda = 10^{-6}$, **69.27**

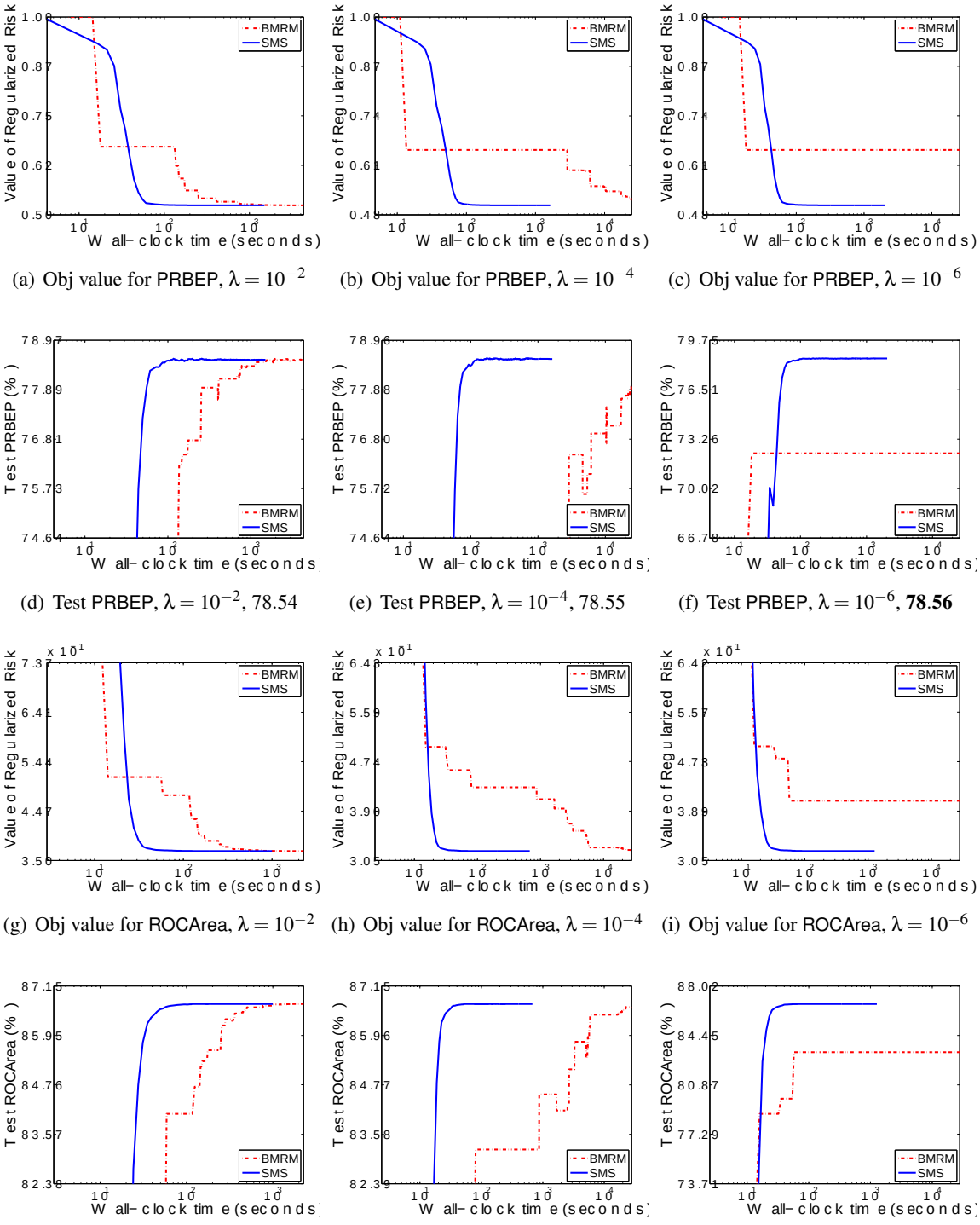


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$



(j) Test ROCArea, $\lambda = 10^{-2}$, 82.64 (k) Test ROCArea, $\lambda = 10^{-4}$, 83.87 (l) Test ROCArea, $\lambda = 10^{-6}$, **84.17**

Figure 7: Results for `covertype`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



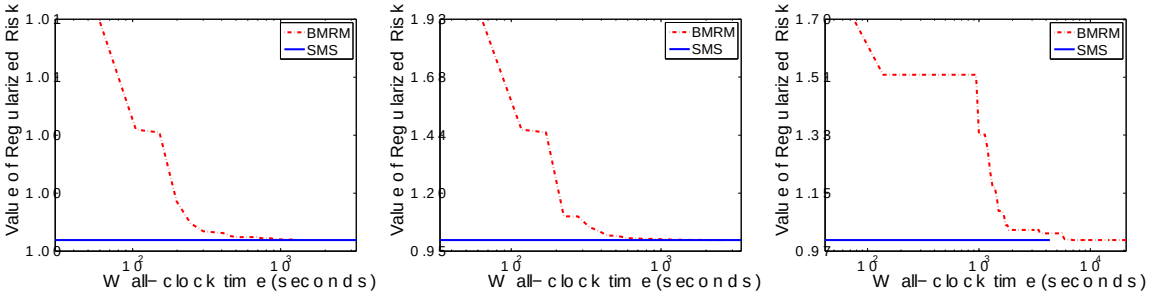
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$

(d) Test PRBEP, $\lambda = 10^{-2}$, 78.54 (e) Test PRBEP, $\lambda = 10^{-4}$, 78.55 (f) Test PRBEP, $\lambda = 10^{-6}$, **78.56**

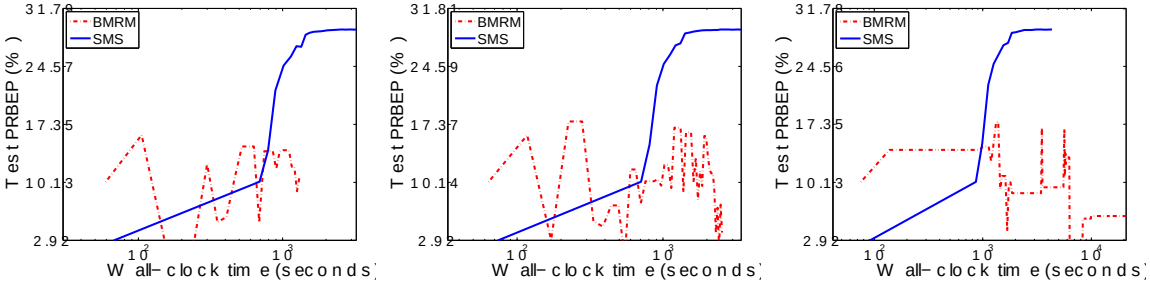
(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

(j) Test ROCArea, $\lambda = 10^{-2}$, 86.71 (k) Test ROCArea, $\lambda = 10^{-4}$, **86.72** (l) Test ROCArea, $\lambda = 10^{-6}$, **86.72**

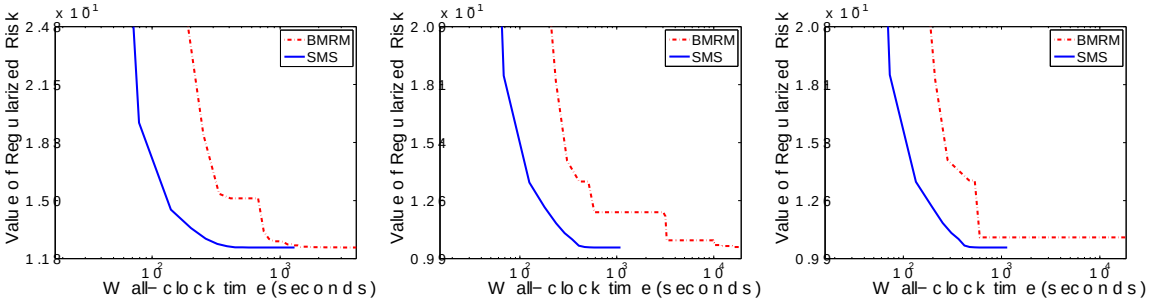
Figure 8: Results for `delta`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



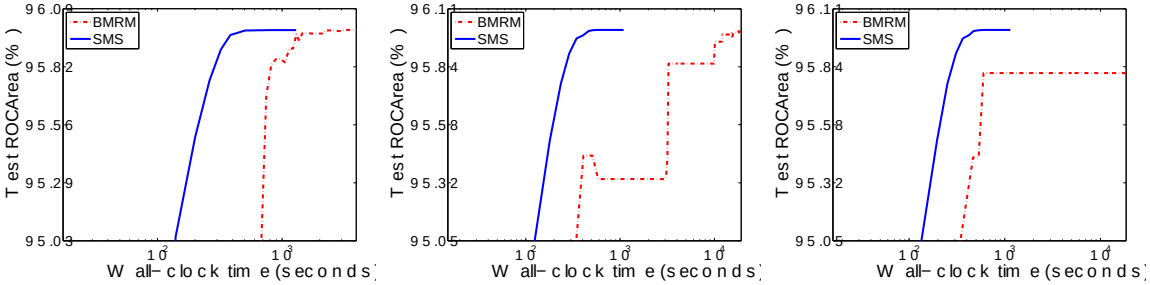
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, **29.14** (e) Test PRBEP, $\lambda = 10^{-4}$, **29.14** (f) Test PRBEP, $\lambda = 10^{-6}$, **29.14**

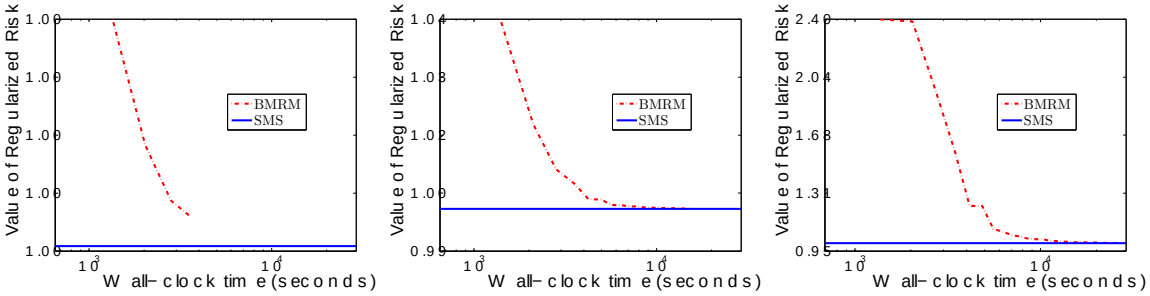


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

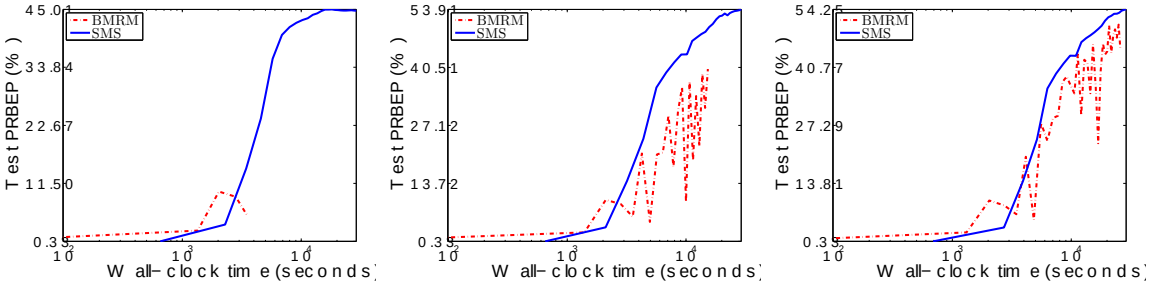


(j) Test ROCArea, $\lambda = 10^{-2}$, 95.99 (k) Test ROCArea, $\lambda = 10^{-4}$, **96.01** (l) Test ROCArea, $\lambda = 10^{-6}$, **96.01**

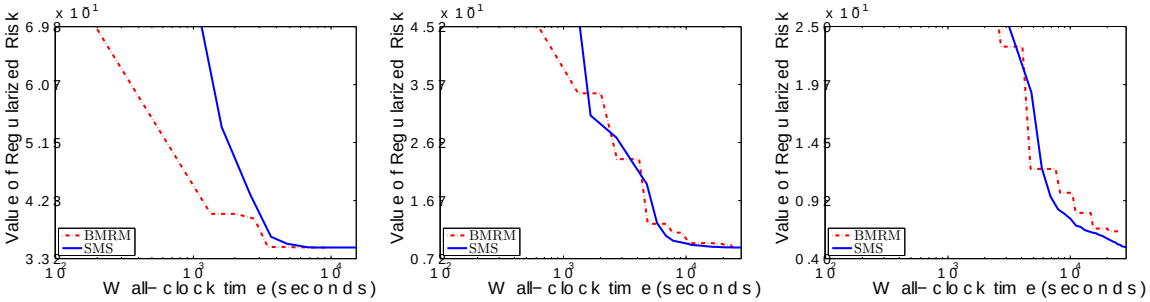
Figure 9: Results for `dna`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions. Both BMRM and our smoothing algorithm are run with 16 processors.



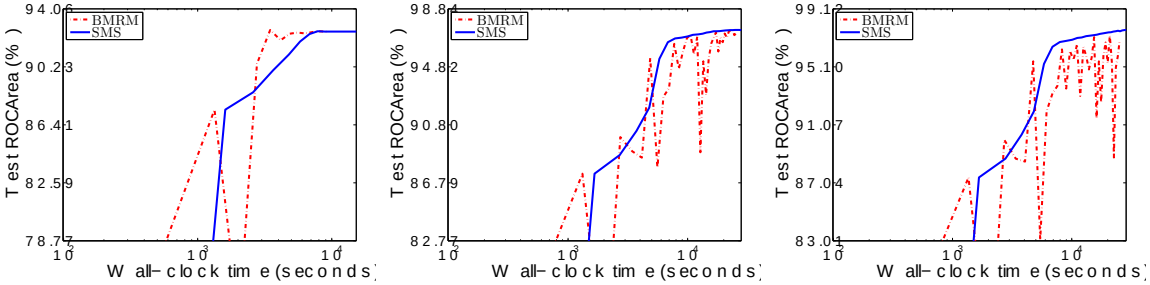
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 45.01 (e) Test PRBEP, $\lambda = 10^{-4}$, 53.91 (f) Test PRBEP, $\lambda = 10^{-6}$, **54.25**

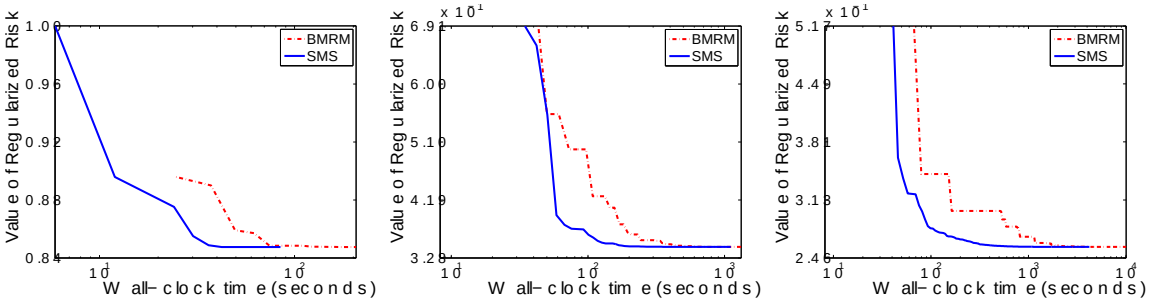


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

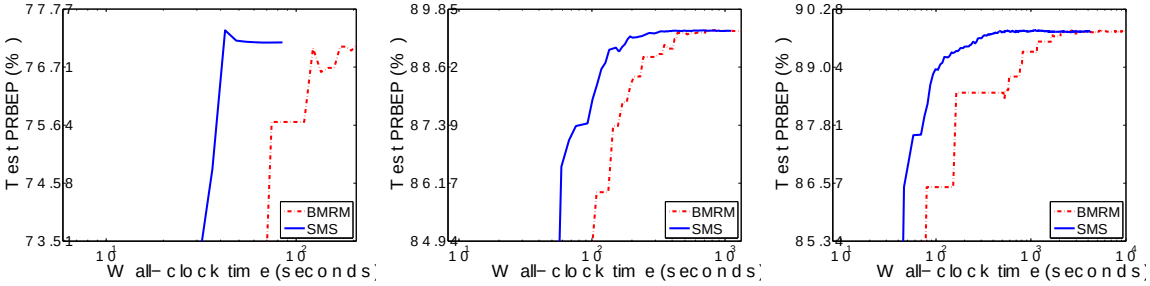


(j) Test ROCArea, $\lambda = 10^{-2}$, 92.57 (k) Test ROCArea, $\lambda = 10^{-4}$, 97.37 (l) Test ROCArea, $\lambda = 10^{-6}$, **97.66**

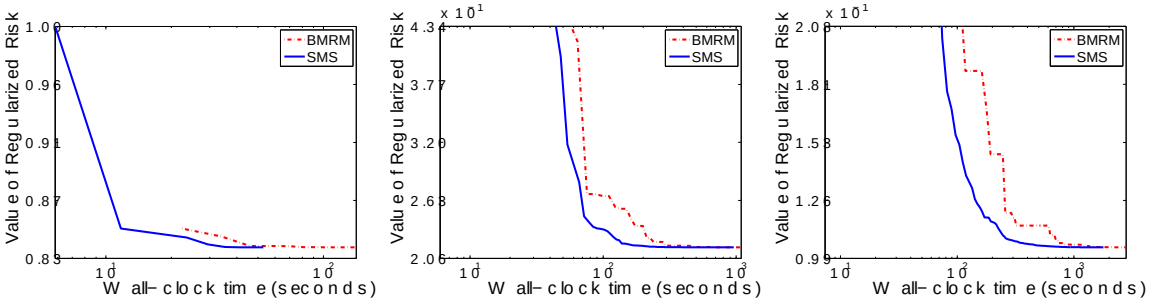
Figure 10: Results for `dna_string`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions. Both BMRM and our smoothing algorithm are run with 32 processors.



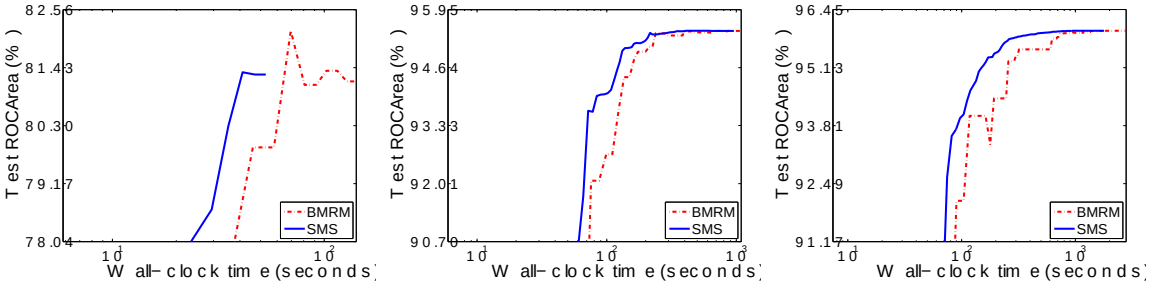
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 77.16 (e) Test PRBEP, $\lambda = 10^{-4}$, 89.39 (f) Test PRBEP, $\lambda = 10^{-6}$, **89.80**

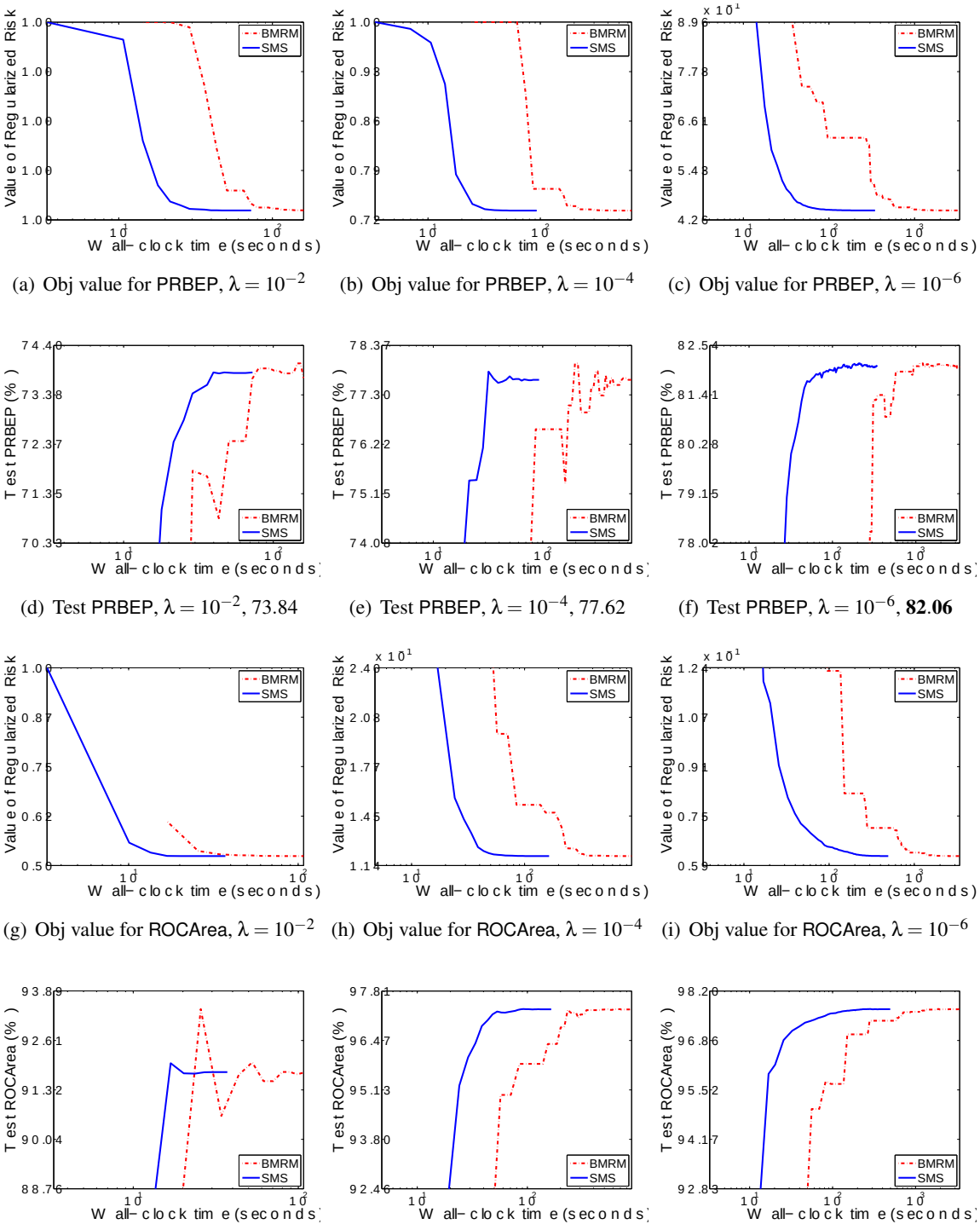


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$



(j) Test ROCArea, $\lambda = 10^{-2}$, 81.30 (k) Test ROCArea, $\lambda = 10^{-4}$, 95.47 (l) Test ROCArea, $\lambda = 10^{-6}$, **95.97**

Figure 11: Results for `epsilon`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



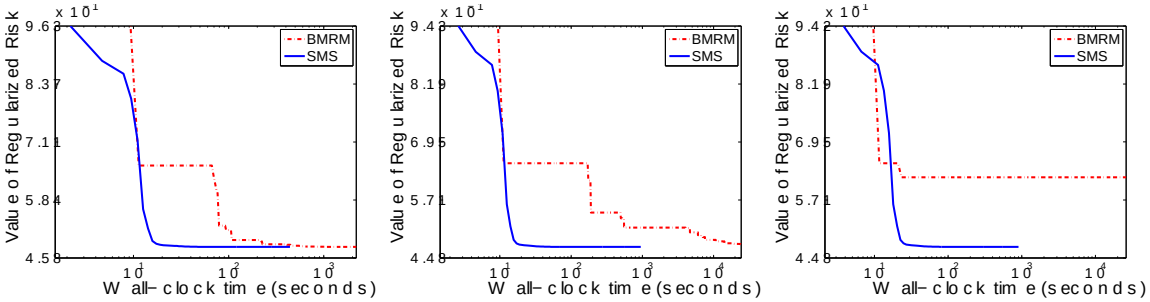
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$

(d) Test PRBEP, $\lambda = 10^{-2}$, 73.84 (e) Test PRBEP, $\lambda = 10^{-4}$, 77.62 (f) Test PRBEP, $\lambda = 10^{-6}$, **82.06**

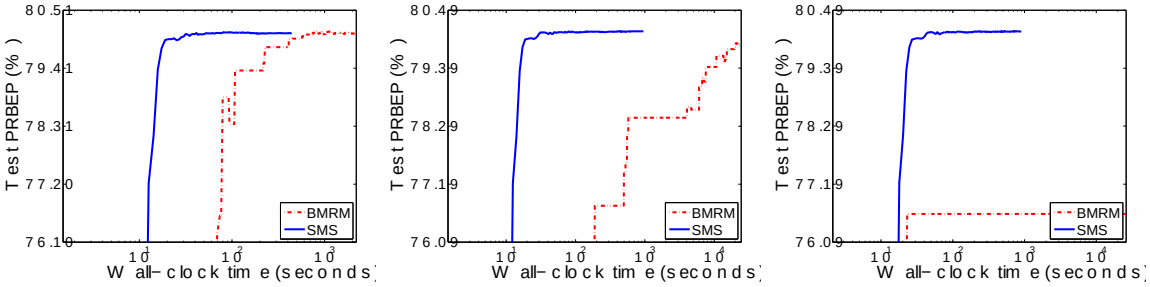
(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

(j) Test ROCArea, $\lambda = 10^{-2}$, 91.79 (k) Test ROCArea, $\lambda = 10^{-4}$, 97.32 (l) Test ROCArea, $\lambda = 10^{-6}$, **97.71**

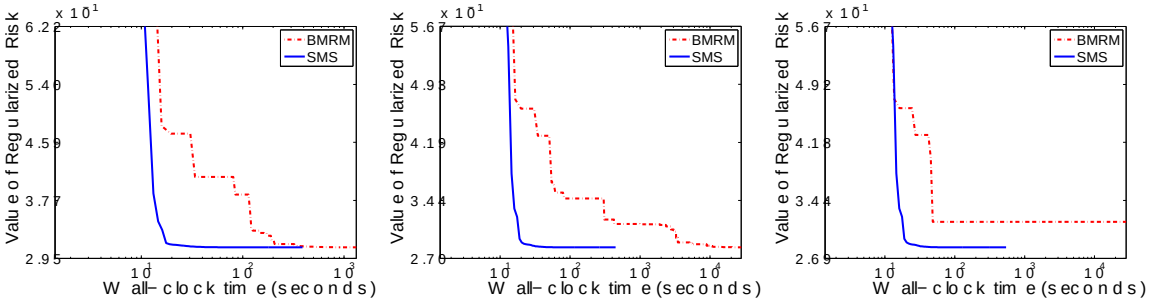
Figure 12: Results for `fd`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



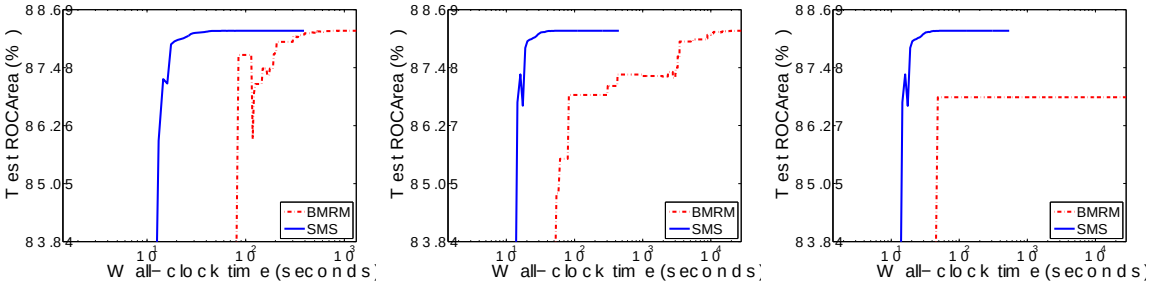
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 80.07 (e) Test PRBEP, $\lambda = 10^{-4}$, **80.09** (f) Test PRBEP, $\lambda = 10^{-6}$, **80.09**

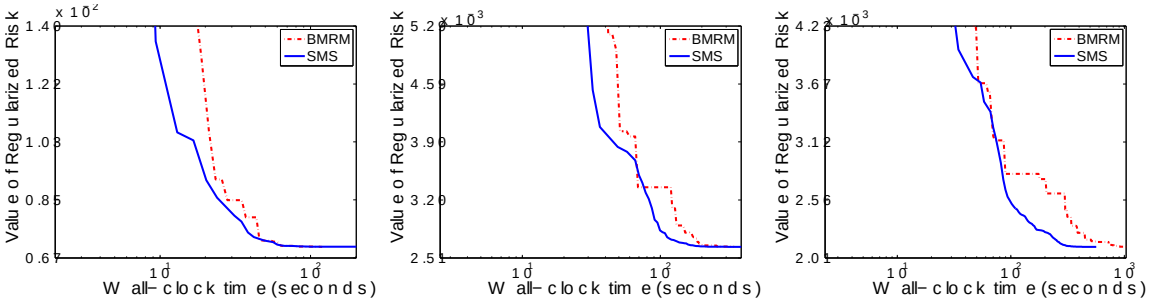


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

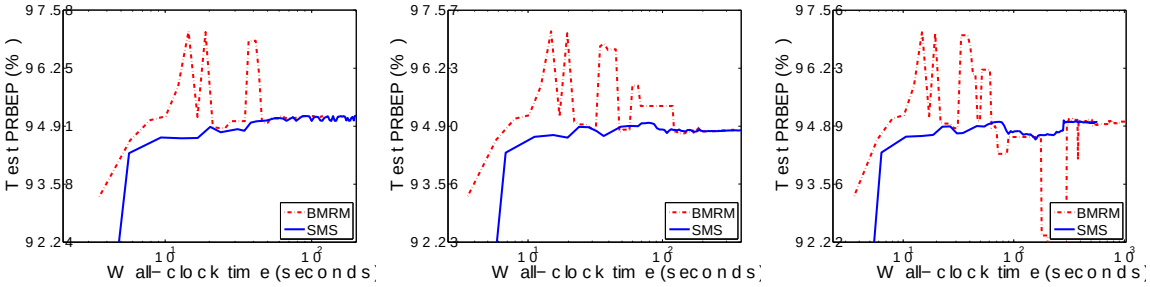


(j) Test ROCArea, $\lambda = 10^{-2}$, **88.25** (k) Test ROCArea, $\lambda = 10^{-4}$, **88.25** (l) Test ROCArea, $\lambda = 10^{-6}$, **88.25**

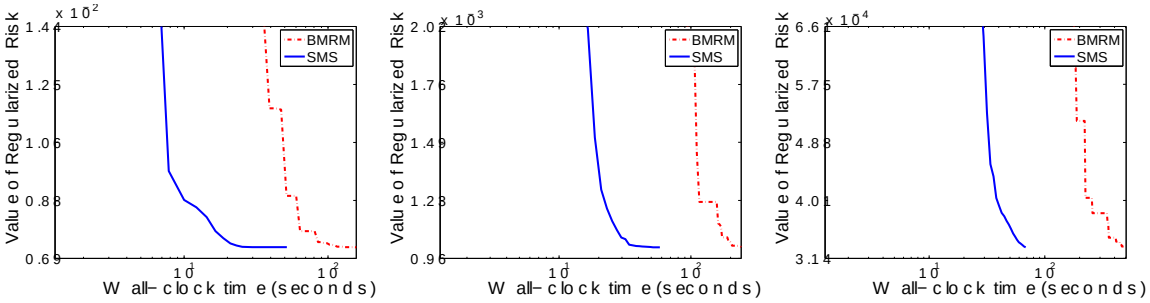
Figure 13: Results for `gamma`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



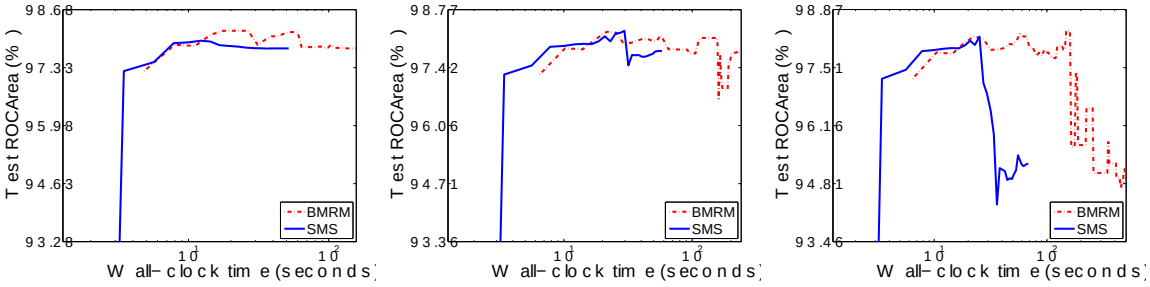
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, **95.07** (e) Test PRBEP, $\lambda = 10^{-4}$, 94.80 (f) Test PRBEP, $\lambda = 10^{-6}$, 94.98

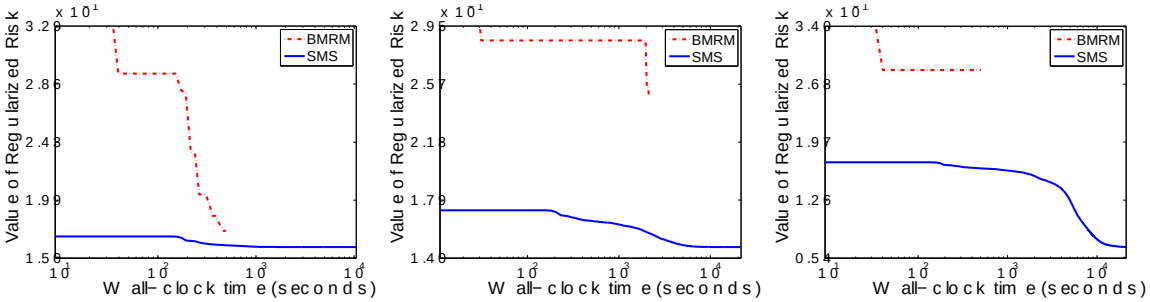


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

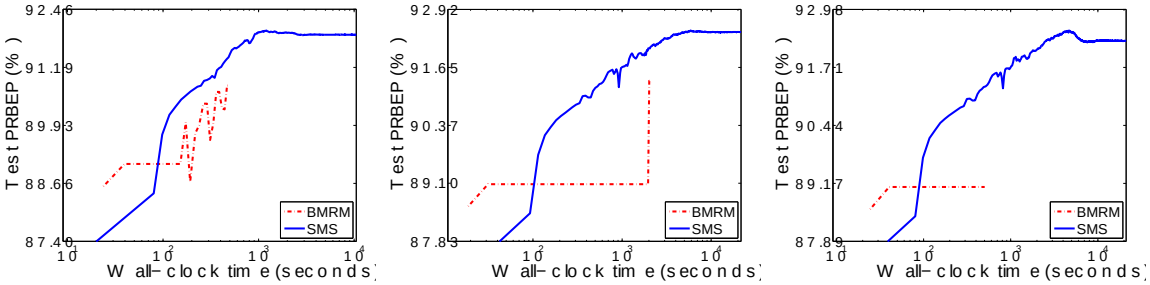


(j) Test ROCArea, $\lambda = 10^{-2}$, 97.78 (k) Test ROCArea, $\lambda = 10^{-4}$, **97.80** (l) Test ROCArea, $\lambda = 10^{-6}$, 95.28

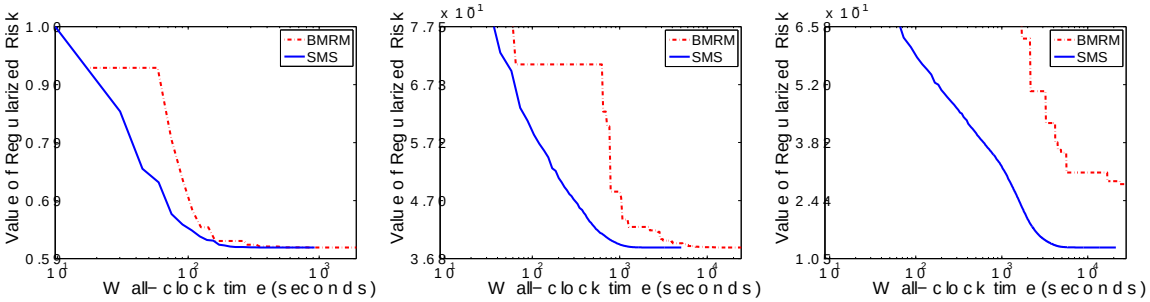
Figure 14: Results for `kdd99`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



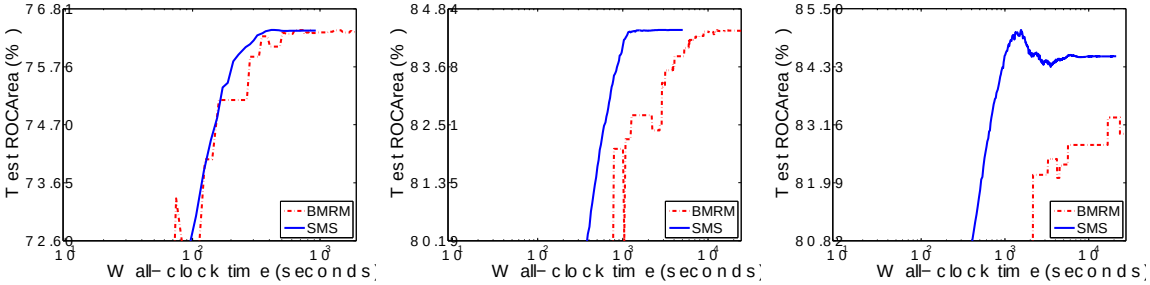
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 91.90 (e) Test PRBEP, $\lambda = 10^{-4}$, 93.02 (f) Test PRBEP, $\lambda = 10^{-6}$, **93.30**

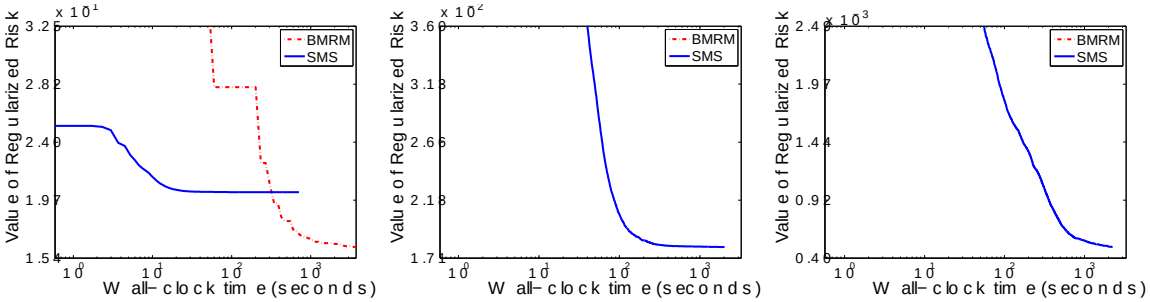


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

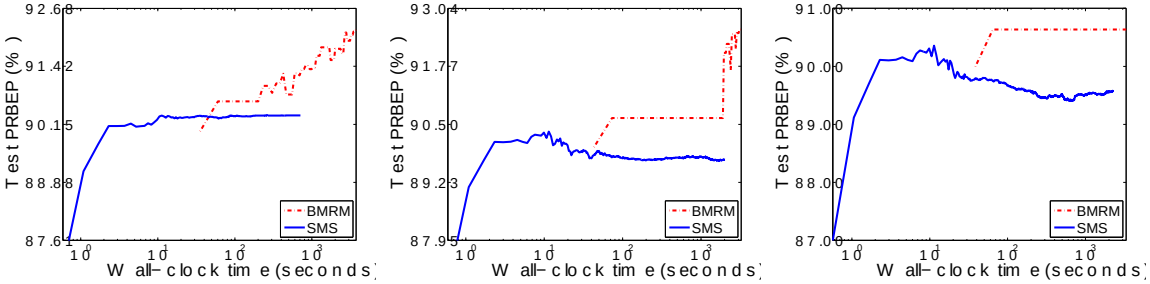


(j) Test ROCArea, $\lambda = 10^{-2}$, 76.41 (k) Test ROCArea, $\lambda = 10^{-4}$, 84.41 (l) Test ROCArea, $\lambda = 10^{-6}$, **84.55**

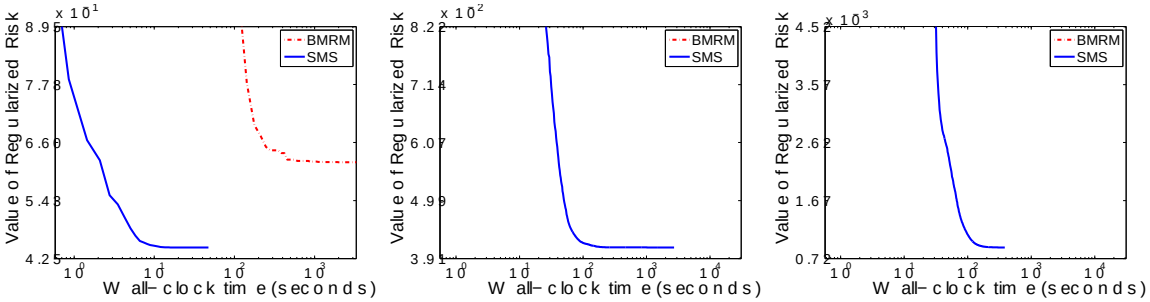
Figure 15: Results for `kdda`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions. In the PRBEP-problem, BMRM fails because the the inner QP solver does not converge even after a large number of iterations.



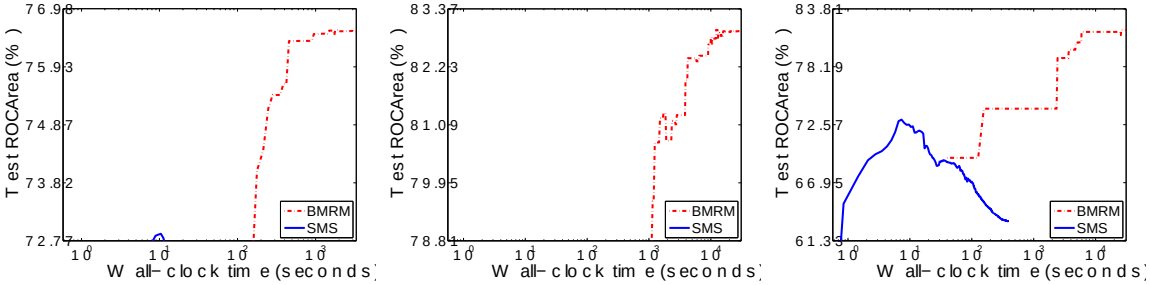
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 92.63 (e) Test PRBEP, $\lambda = 10^{-4}$, 93.02 (f) Test PRBEP, $\lambda = 10^{-6}$, **93.30**

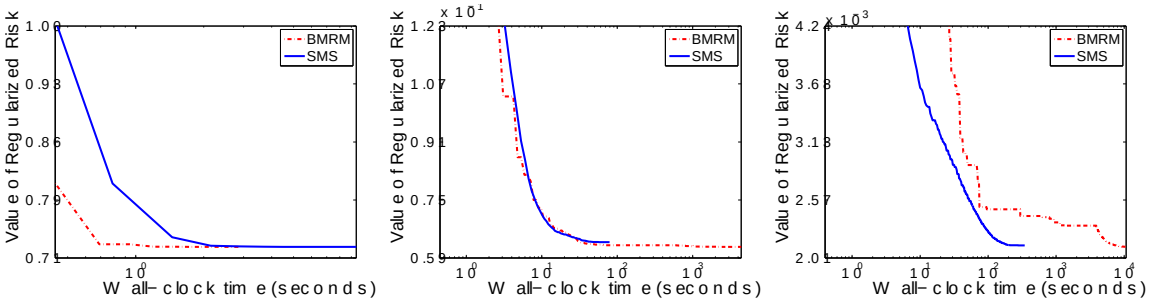


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

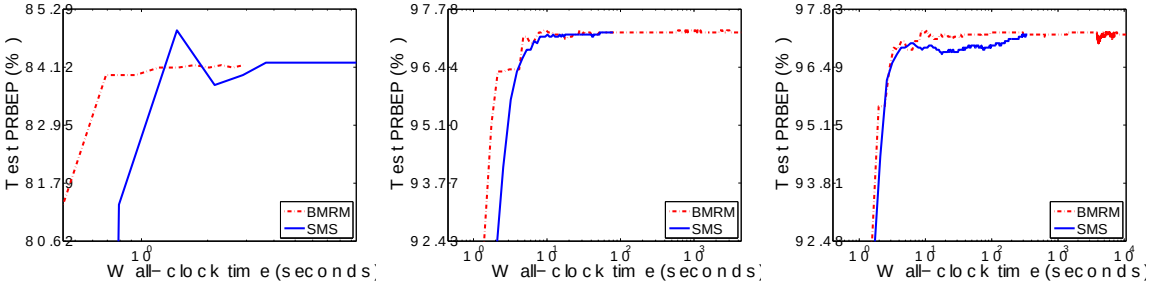


(j) Test ROCArea, $\lambda = 10^{-2}$, 76.58 (k) Test ROCArea, $\lambda = 10^{-4}$, 82.95 (l) Test ROCArea, $\lambda = 10^{-6}$, **85.29**

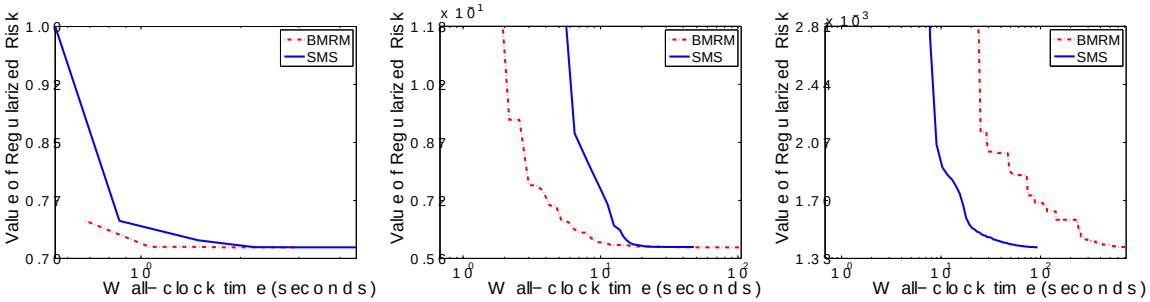
Figure 16: Results for kddb. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions. In the PRBEP-problem, BMRM fails because the inner QP solver does not converge even after a large number of iterations.



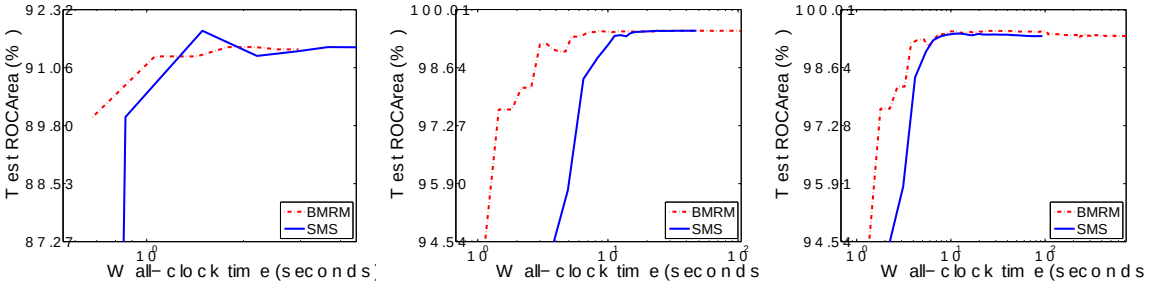
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 84.21 (e) Test PRBEP, $\lambda = 10^{-4}$, **97.24** (f) Test PRBEP, $\lambda = 10^{-6}$, **97.24**

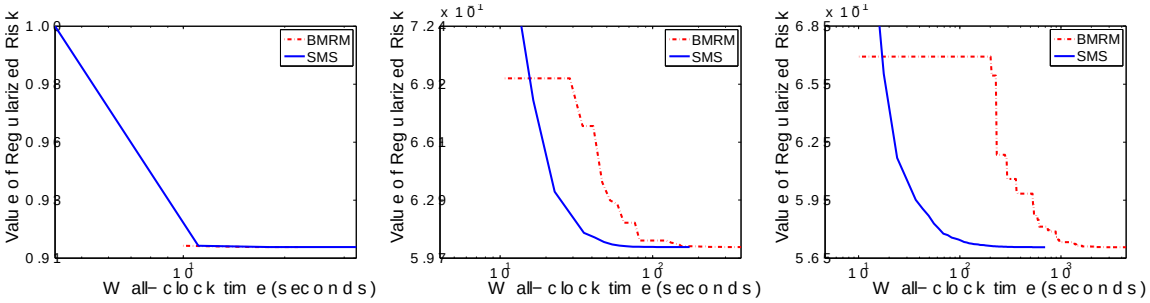


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

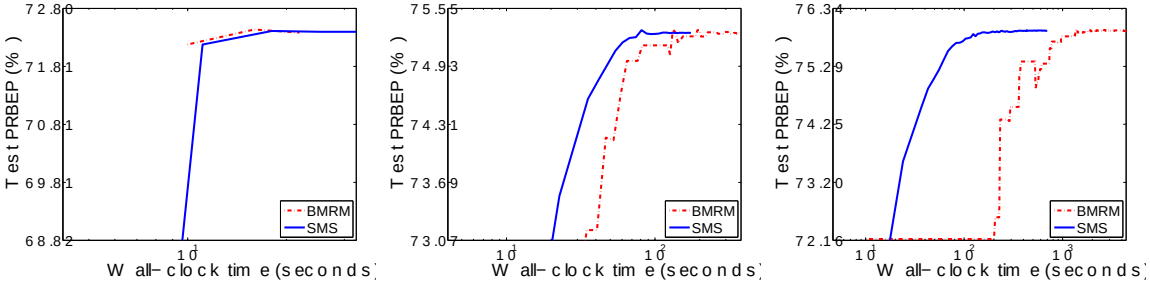


(j) Test ROCArea, $\lambda = 10^{-2}$, 91.51 (k) Test ROCArea, $\lambda = 10^{-4}$, **99.51** (l) Test ROCArea, $\lambda = 10^{-6}$, 99.39

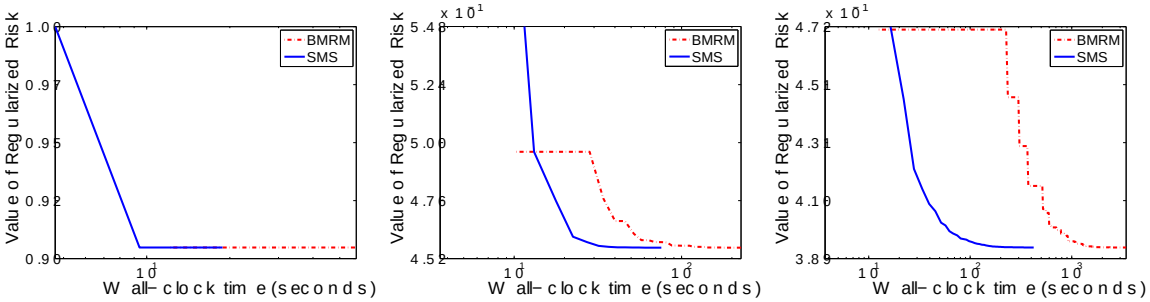
Figure 17: Results for `news20`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



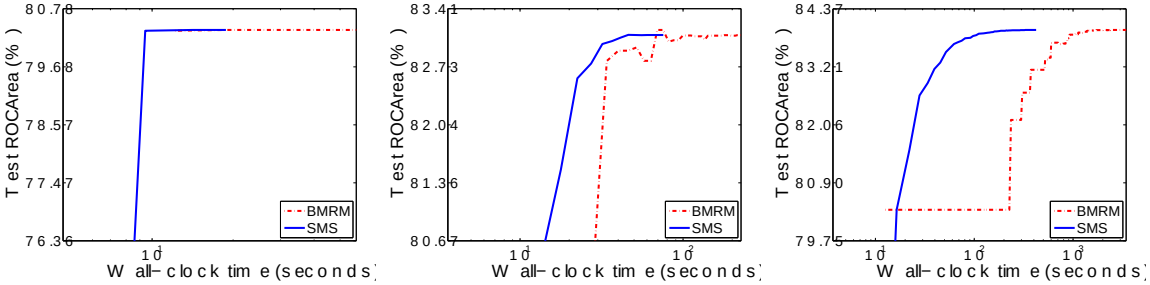
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 72.40 (e) Test PRBEP, $\lambda = 10^{-flff4}$, 75.29 (f) Test PRBEP, $\lambda = 10^{-6}$, **75.93**

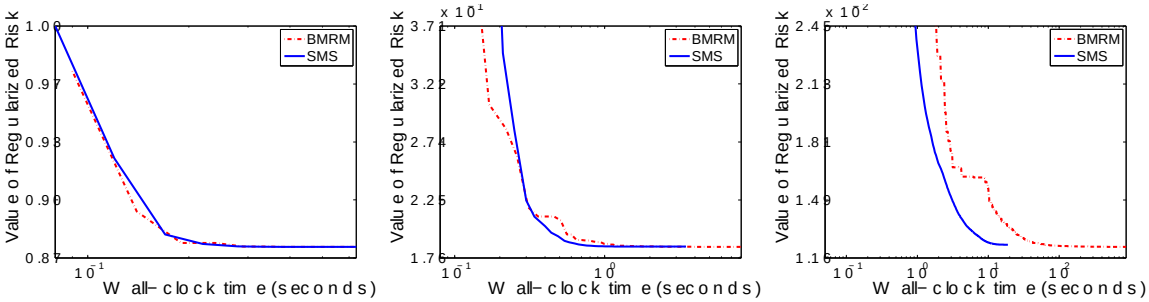


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

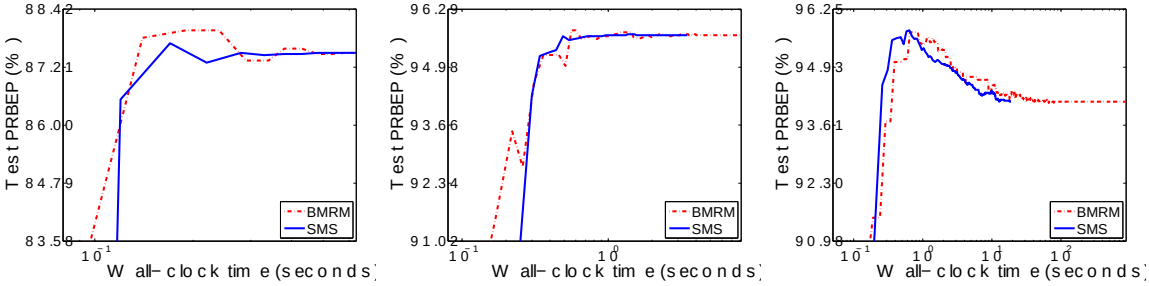


(j) Test ROCArea, $\lambda = 10^{-2}$, 80.38 (k) Test ROCArea, $\lambda = 10^{-4}$, 83.10 (l) Test ROCArea, $\lambda = 10^{-6}$, **83.94**

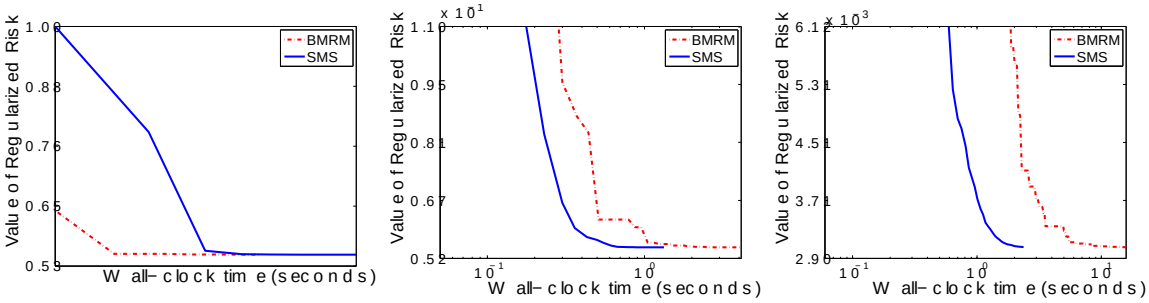
Figure 18: Results for `ocr`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions. Both BMRM and our smoothing algorithm are run with 16 processors.



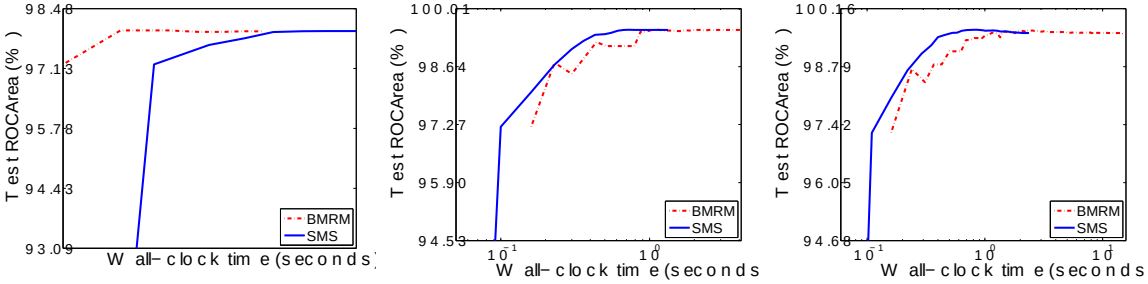
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 87.51 (e) Test PRBEP, $\lambda = 10^{-4}$, **95.70** (f) Test PRBEP, $\lambda = 10^{-6}$, 94.14

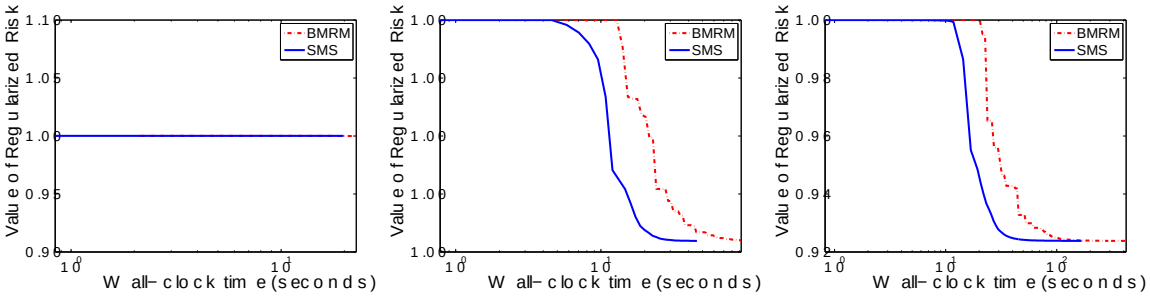


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

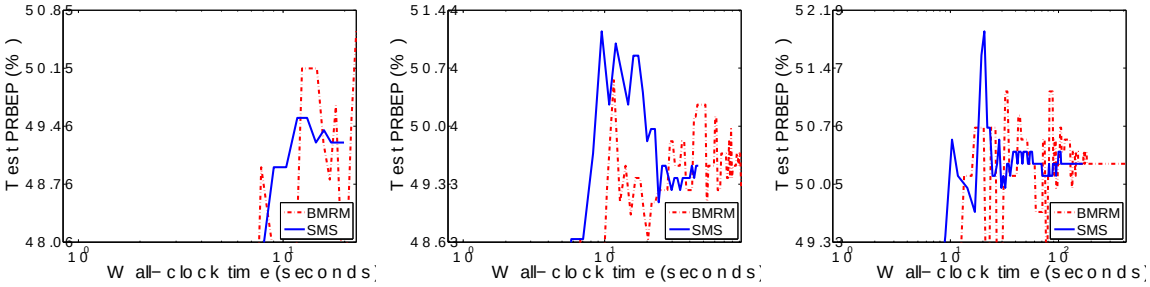


(j) Test ROCArea, $\lambda = 10^{-2}$, 97.97 (k) Test ROCArea, $\lambda = 10^{-4}$, 99.51 (l) Test ROCArea, $\lambda = 10^{-6}$, **99.58**

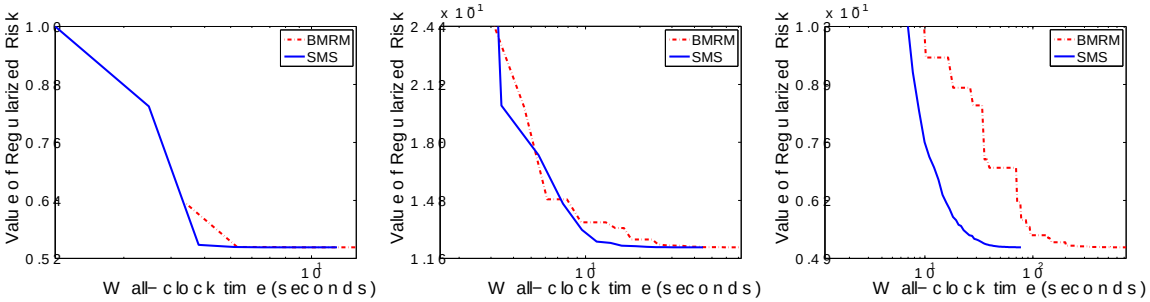
Figure 19: Results for `real-sim`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



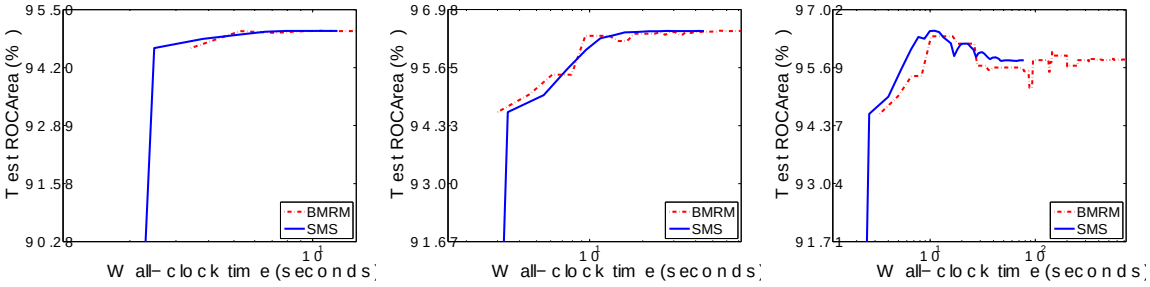
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 49.26 (e) Test PRBEP, $\lambda = 10^{-4}$, 49.55 (f) Test PRBEP, $\lambda = 10^{-6}$, **50.30**

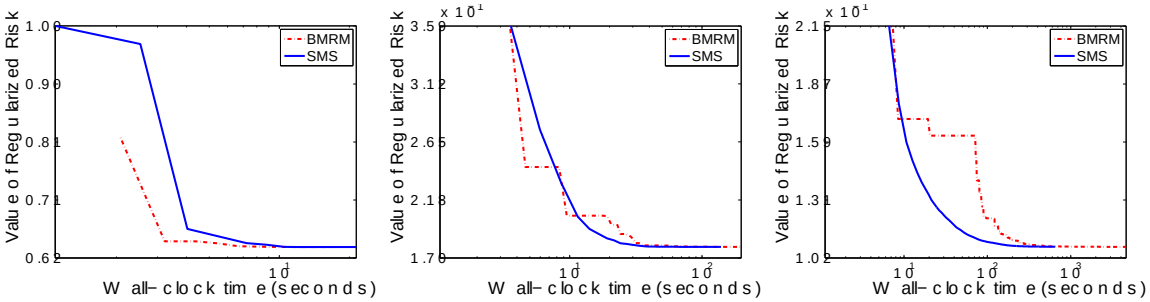


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

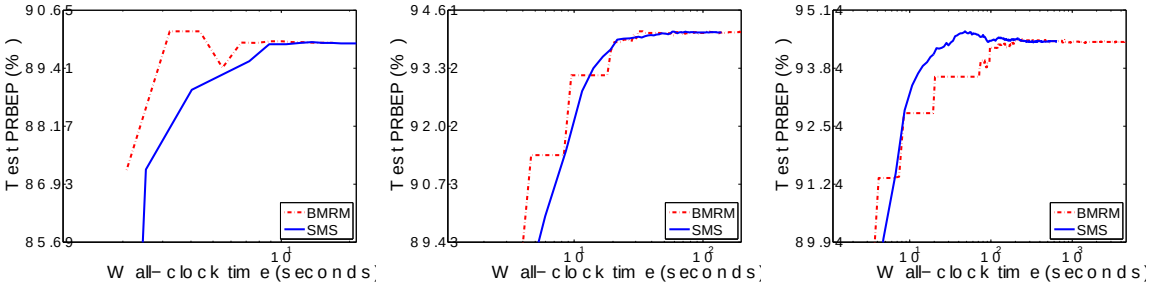


(j) Test ROCArea, $\lambda = 10^{-2}$, 95.02 (k) Test ROCArea, $\lambda = 10^{-4}$, **96.49** (l) Test ROCArea, $\lambda = 10^{-6}$, 95.86

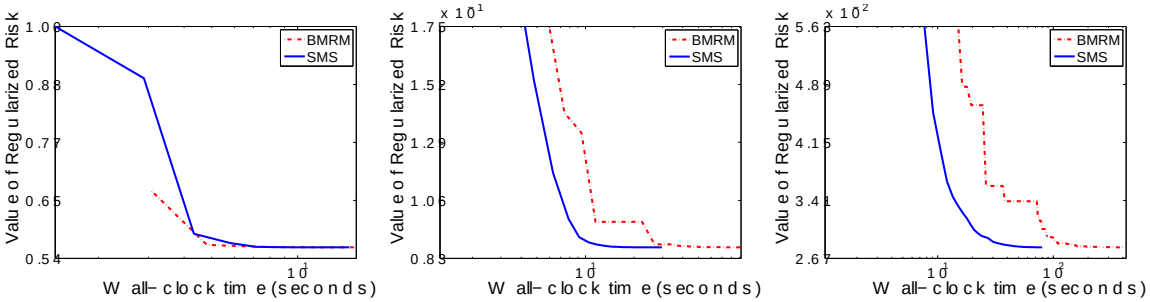
Figure 20: Results for `reuters-c11`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



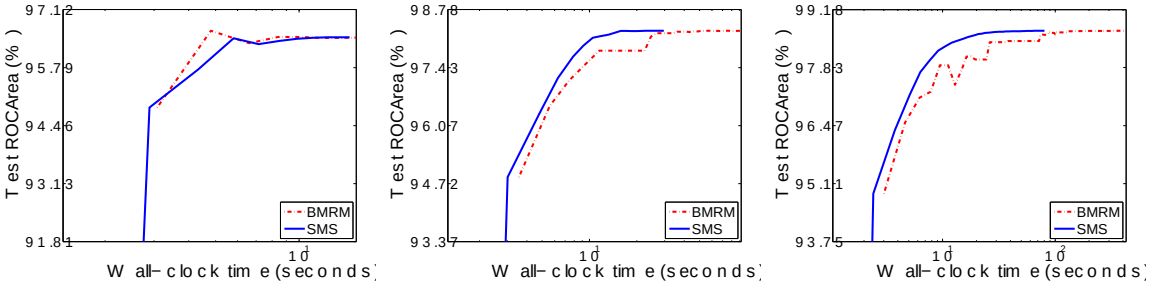
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 89.94 (e) Test PRBEP, $\lambda = 10^{-4}$, 94.11 (f) Test PRBEP, $\lambda = 10^{-6}$, **94.44**

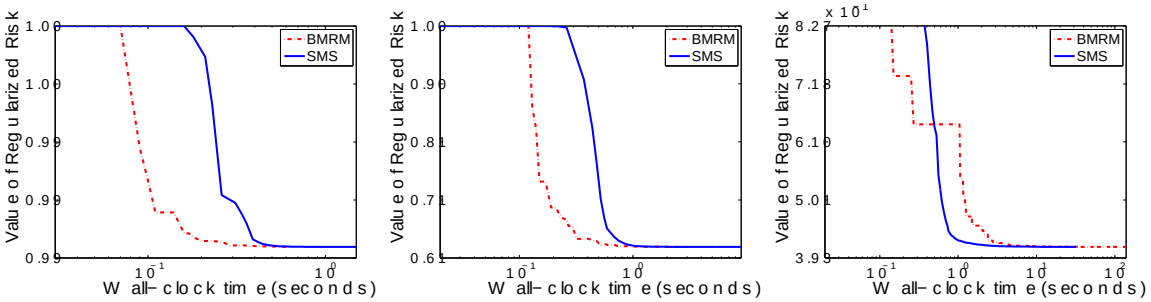


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

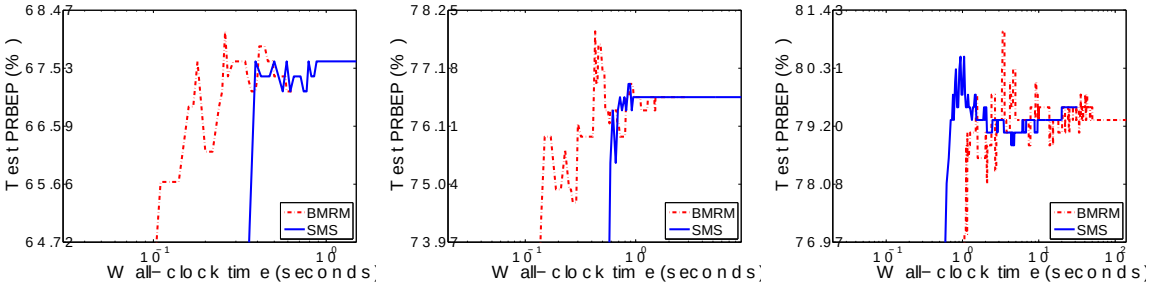


(j) Test ROCArea, $\lambda = 10^{-2}$, 96.49 (k) Test ROCArea, $\lambda = 10^{-4}$, 98.28 (l) Test ROCArea, $\lambda = 10^{-6}$, **98.69**

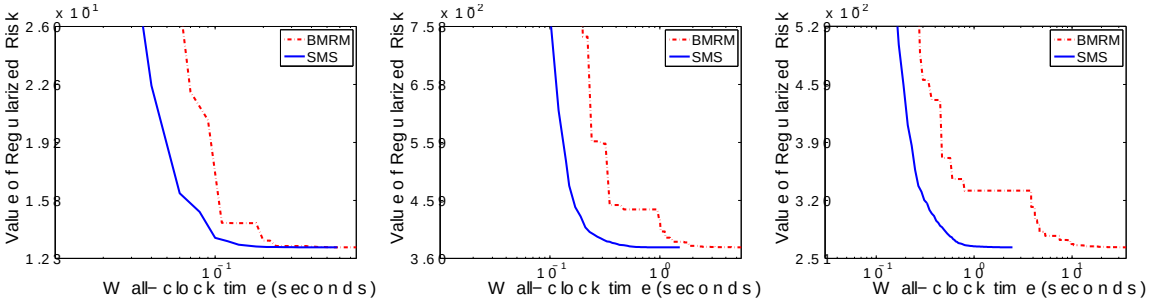
Figure 21: Results for `reuters-ccat`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



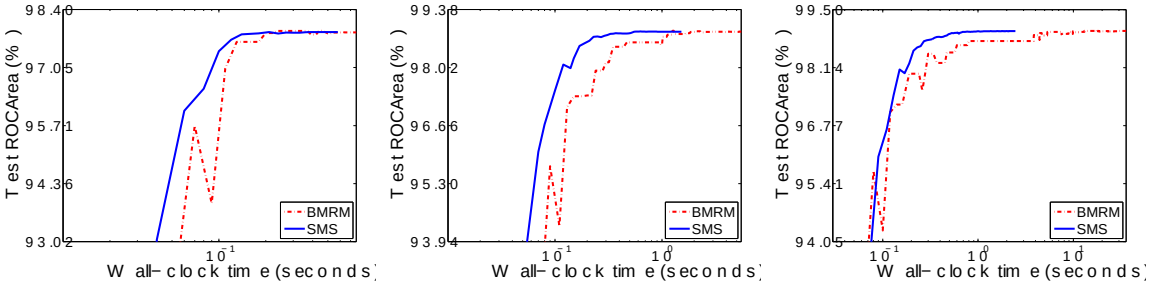
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 67.64 (e) Test PRBEP, $\lambda = 10^{-4}$, 76.64 (f) Test PRBEP, $\lambda = 10^{-6}$, **79.56**

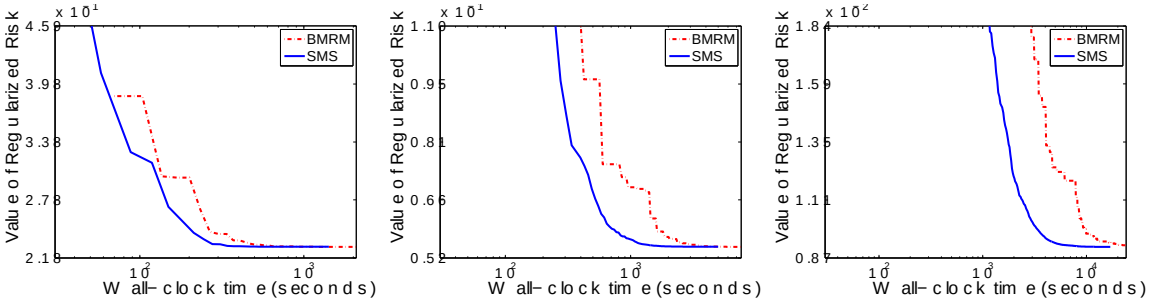


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

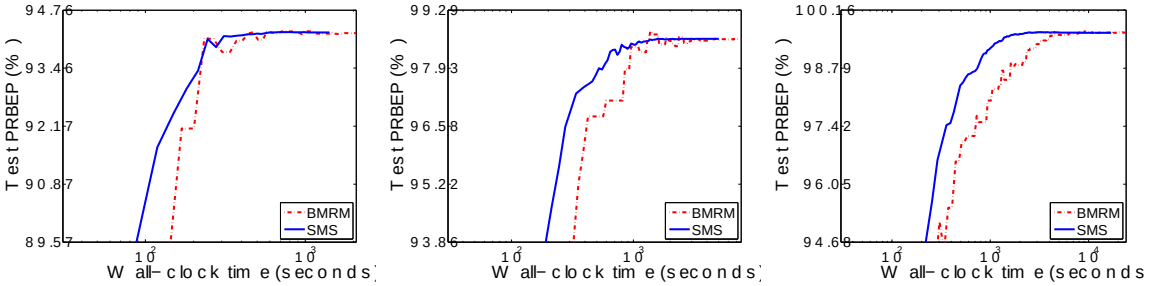


(j) Test ROCArea, $\lambda = 10^{-2}$, 97.88 (k) Test ROCArea, $\lambda = 10^{-4}$, 98.86 (l) Test ROCArea, $\lambda = 10^{-6}$, **98.99**

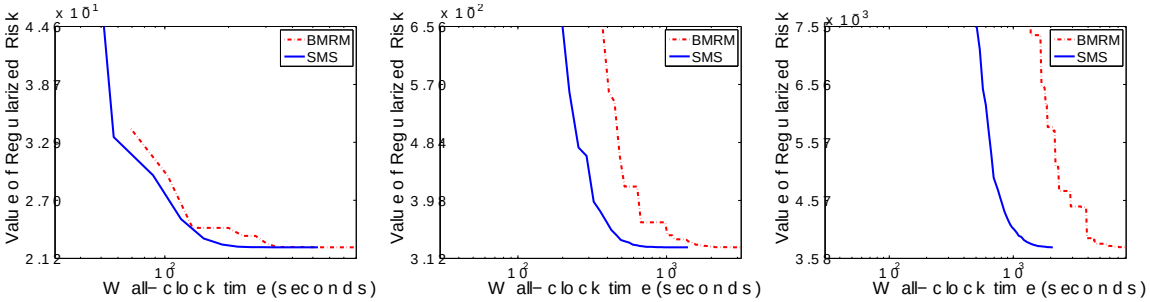
Figure 22: Results for `web8`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



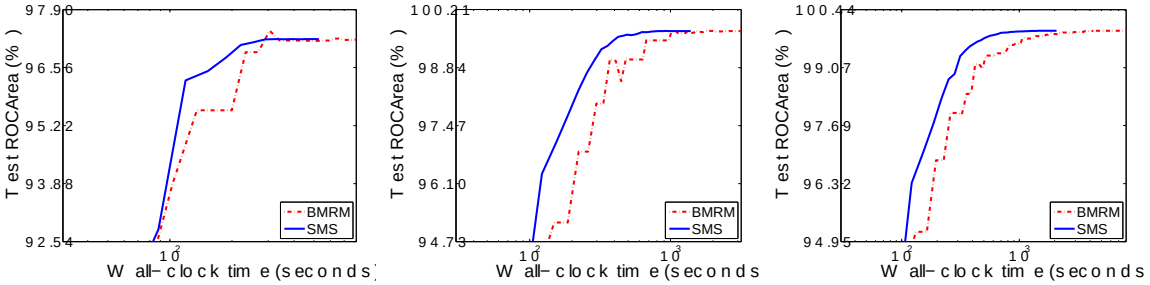
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 94.26 (e) Test PRBEP, $\lambda = 10^{-4}$, 98.62 (f) Test PRBEP, $\lambda = 10^{-6}$, **99.62**

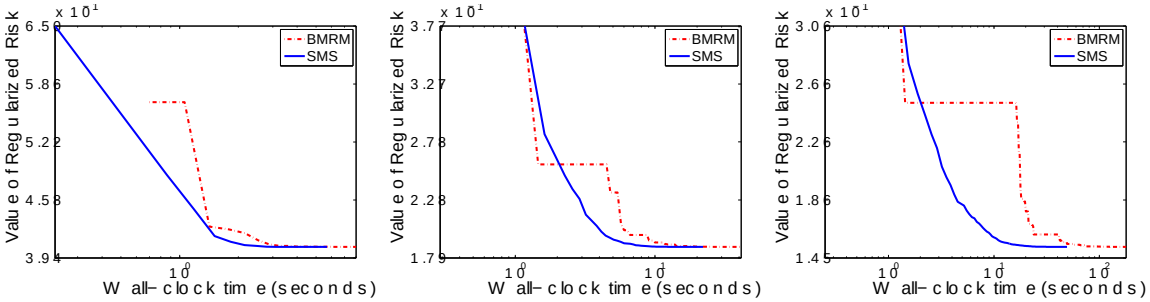


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

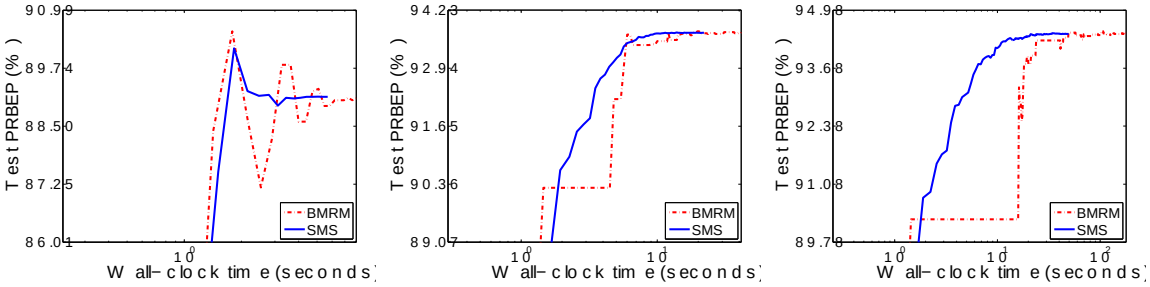


(j) Test ROCArea, $\lambda = 10^{-2}$, 97.21 (k) Test ROCArea, $\lambda = 10^{-4}$, 99.71 (l) Test ROCArea, $\lambda = 10^{-6}$, **99.94**

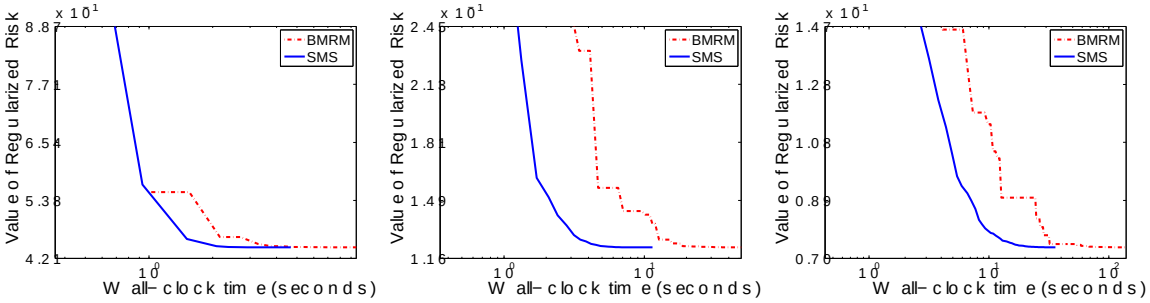
Figure 23: Results for `webspam-t`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



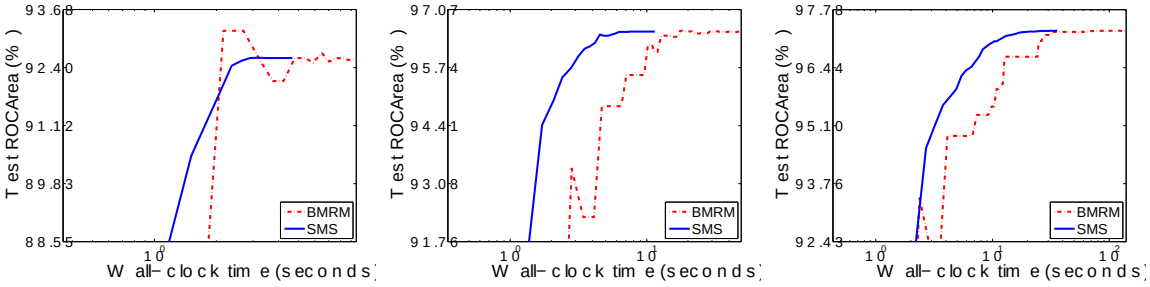
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 89.13 (e) Test PRBEP, $\lambda = 10^{-4}$, 93.73 (f) Test PRBEP, $\lambda = 10^{-6}$, **94.45**

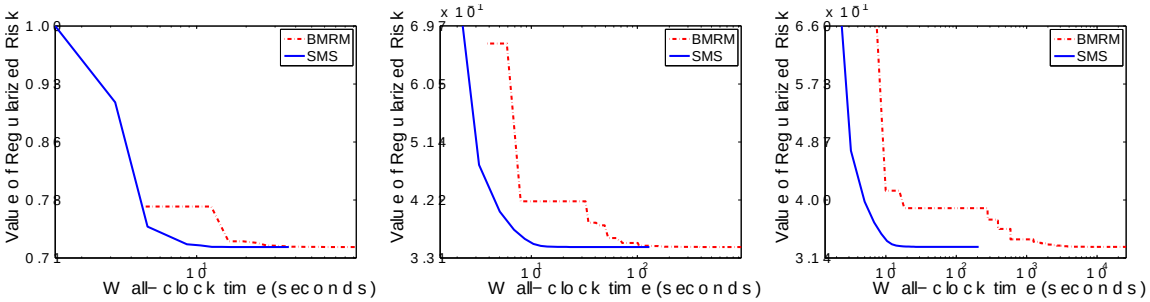


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

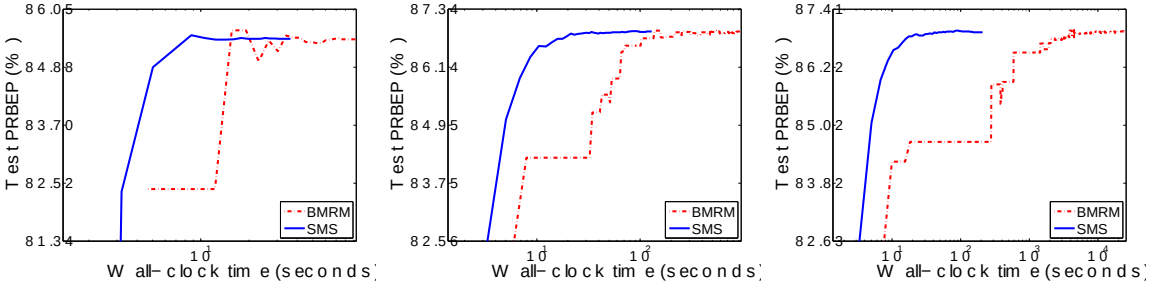


(j) Test ROCArea, $\lambda = 10^{-2}$, 92.61 (k) Test ROCArea, $\lambda = 10^{-4}$, 96.56 (l) Test ROCArea, $\lambda = 10^{-6}$, **97.29**

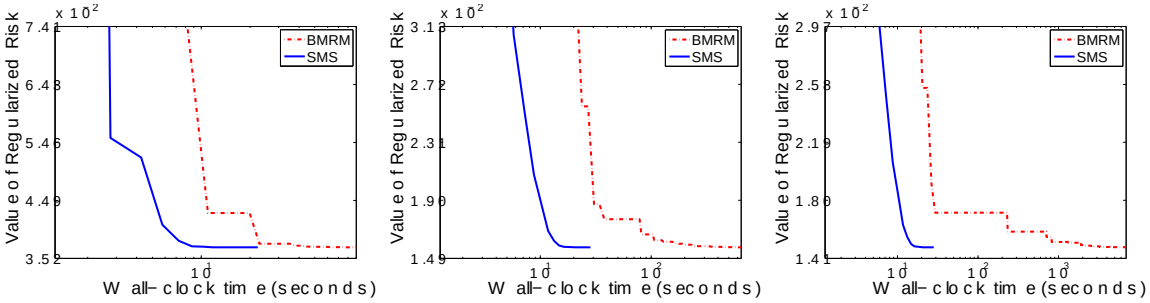
Figure 24: Results for `webspam-u`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



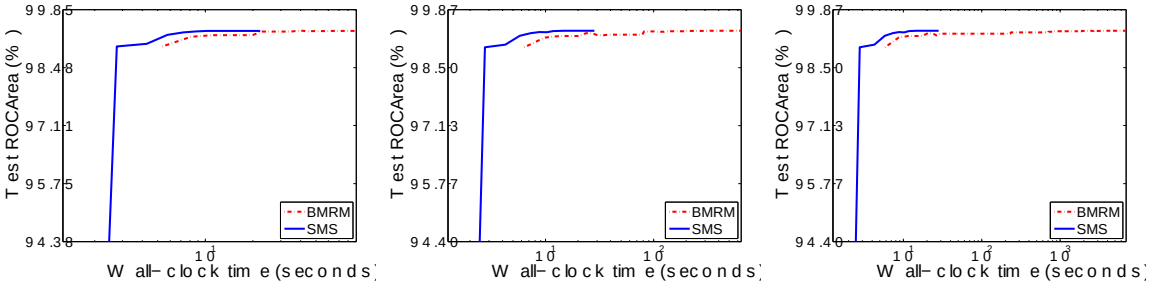
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 85.45 (e) Test PRBEP, $\lambda = 10^{-4}$, 86.88 (f) Test PRBEP, $\lambda = 10^{-6}$, **86.94**

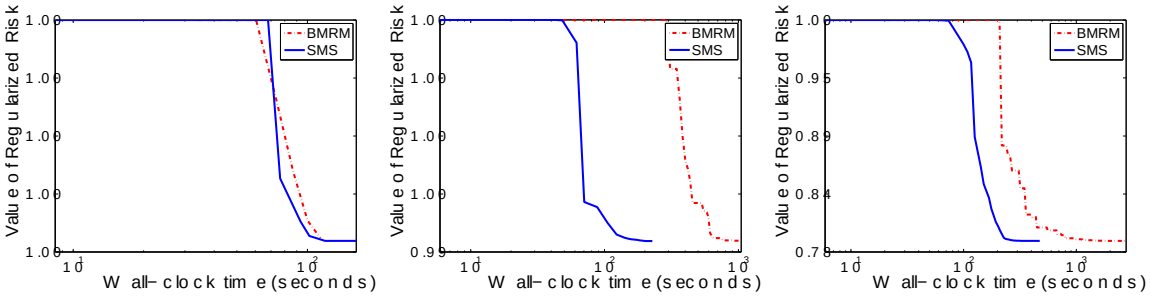


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$

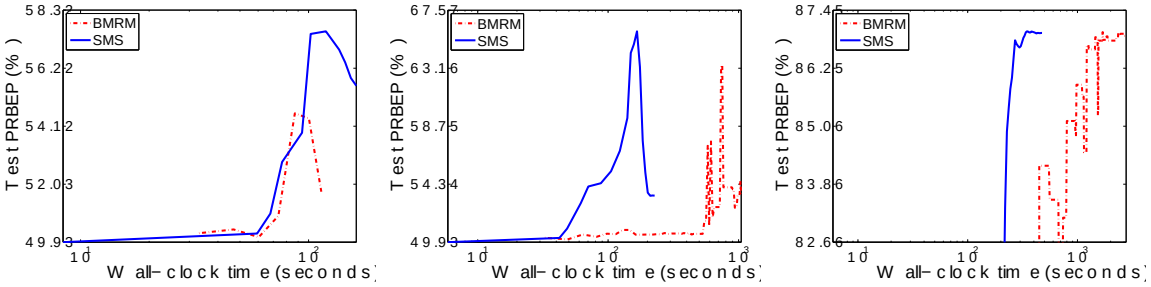


(j) Test ROCArea, $\lambda = 10^{-2}$, 99.34 (k) Test ROCArea, $\lambda = 10^{-4}$, **99.37** (l) Test ROCArea, $\lambda = 10^{-6}$, **99.37**

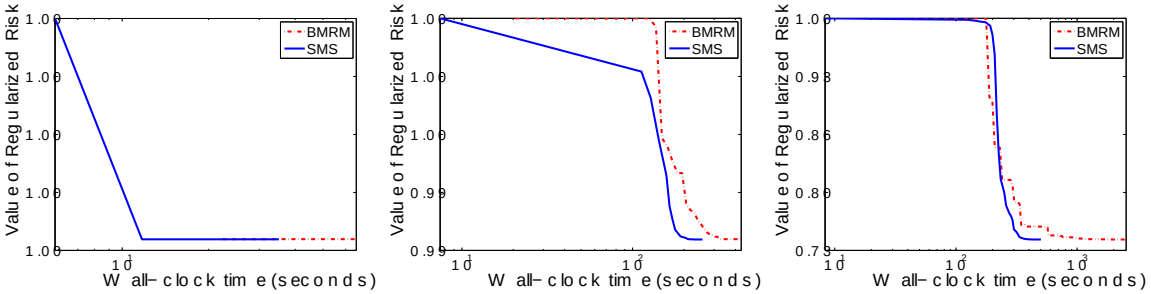
Figure 25: Results for `worm`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



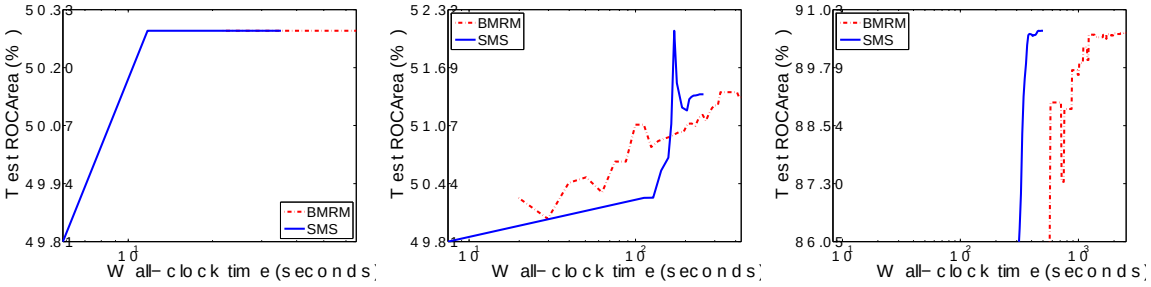
(a) Obj value for PRBEP, $\lambda = 10^{-2}$ (b) Obj value for PRBEP, $\lambda = 10^{-4}$ (c) Obj value for PRBEP, $\lambda = 10^{-6}$



(d) Test PRBEP, $\lambda = 10^{-2}$, 55.59 (e) Test PRBEP, $\lambda = 10^{-4}$, 53.47 (f) Test PRBEP, $\lambda = 10^{-6}$, **86.98**

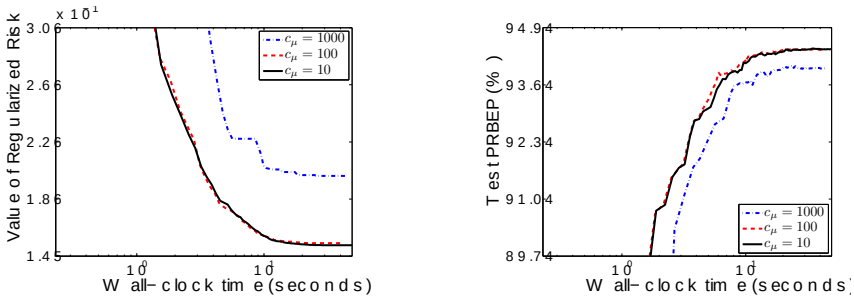


(g) Obj value for ROCArea, $\lambda = 10^{-2}$ (h) Obj value for ROCArea, $\lambda = 10^{-4}$ (i) Obj value for ROCArea, $\lambda = 10^{-6}$



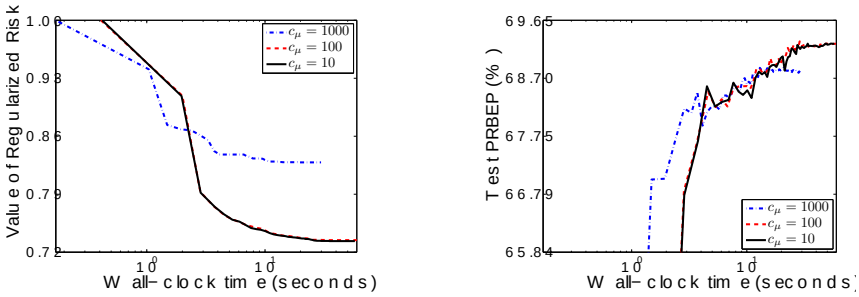
(j) Test ROCArea, $\lambda = 10^{-2}$, 50.28 (k) Test ROCArea, $\lambda = 10^{-4}$, 51.41 (l) Test ROCArea, $\lambda = 10^{-6}$, **90.58**

Figure 26: Results for `zeta`. The optimal test performance among all values of λ is highlighted in boldface in the sub-figure captions.



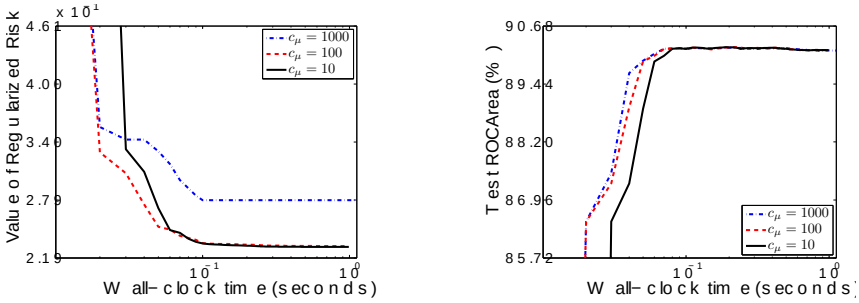
(a) Obj value for PRBEP (b) Test PRBEP

Figure 27: c_μ test: `webspam-u`



(a) Obj value for PRBEP (b) Test PRBEP

Figure 28: c_μ test: `covertype`

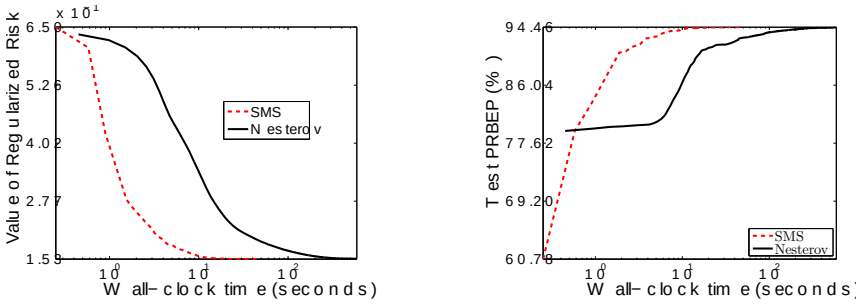


(a) Obj value for ROCArea (b) Test ROCArea

Figure 29: c_μ test: `adult9`

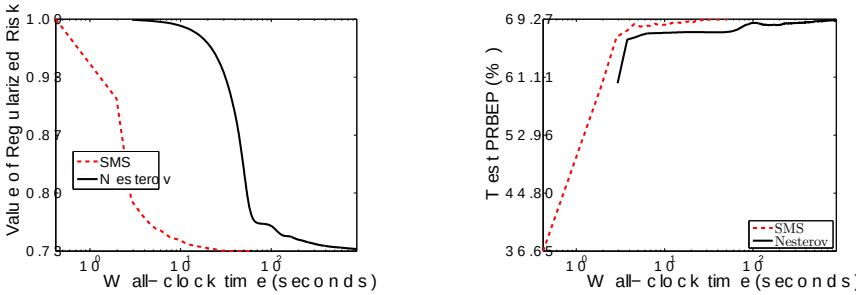
it is easy to smooth the L_1 norm regularizer, it is not recommended; the sparsity of the solution is an important statistical property of these algorithms and smoothing destroys this property.

In future work we would like to extend our techniques to handle more complicated contingency based multivariate performance measures such as the F_1-score. We would also like to extend smoothing to matching loss functions commonly used in ranking, where we believe our techniques will solve a smoothed version of the Hungarian marriage problem.



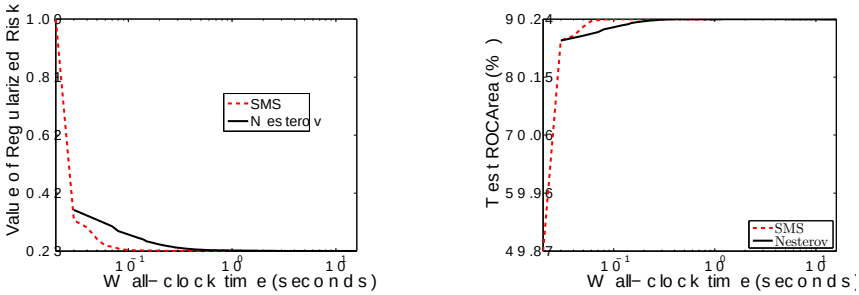
(a) Obj value for PRBEP (b) Test PRBEP

Figure 30: SMS vs AGM: `webspam-u`



(a) Obj value for PRBEP (b) Test PRBEP

Figure 31: SMS vs AGM: `covertype`



(a) Obj value for ROCArea (b) Test ROCArea

Figure 32: SMS vs AGM: `adult9`

Acknowledgments

We thank the PETSc and TAO developers for making available their code, updating their software based on our feature requests, and patiently answering all our questions. We thank the anonymous reviewers of the earlier versions of this paper for their helpful comments and suggestions. Xinhua Zhang would like to acknowledge support from the Alberta Innovates Centre for Machine Learning. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program. Ankan Saha would like to acknowledge the fellowship support from the department of Computer Science at the University of Chicago. The work of S.V.N. Vishwanathan is partially supported by a grant from Google and NSF grant IIS-1117705.

Appendix A. The Smoothing Procedure

The idea of the smoothing technique in Nesterov (2005) can be motivated by using the Theorem 4.2.1 and 4.2.2 in Hiriart-Urruty and Lemaréchal (1996).

Lemma 9 *If $f:\mathbb{R}^n\to\mathbb{R}$ is convex and differentiable, and ∇f is Lipschitz continuous with constant L (called L-l.c.g), then $f^\star$ is strongly convex with modulus $\frac{1}{L}$ (called $\frac{1}{L}$-sc). Conversely, if $f:\mathbb{R}^n\to\mathbb{R}\cup\{\infty\}$ is σ-sc, then $f^\star$ is finite on $\mathbb{R}^n$ and is $\frac{1}{\sigma}$-l.c.g.*

Since $g+\mu d$ is μ-sc, Lemma 9 implies $g_\mu^\star$ is $\frac{1}{\mu}$-*l.c.g*. By chain rule, one can show that $g_\mu^\star(A^\top\mathbf{w})$ is L_μ-*l.c.g* where $L_\mu\le\frac{1}{\mu}\|A\|^2$. Further, the definition of Fenchel dual implies the following uniform deviation bound:

$$g_\mu^\star(\boldsymbol{u})=\max_{\boldsymbol{\alpha}\in Q}\{\langle\boldsymbol{\alpha},\boldsymbol{u}\rangle-g(\boldsymbol{\alpha})-\mu d(\boldsymbol{\alpha})\}\begin{cases}\le\max\limits_{\boldsymbol{\alpha}\in Q}\{\langle\boldsymbol{\alpha},\boldsymbol{u}\rangle-g(\boldsymbol{\alpha})\}=g^\star(\boldsymbol{u})\\ \ge\max\limits_{\boldsymbol{\alpha}\in Q}\{\langle\boldsymbol{\alpha},\boldsymbol{u}\rangle-g(\boldsymbol{\alpha})-\mu D\}=g^\star(\boldsymbol{u})-\mu D\end{cases}$$

$$\implies\qquad g^\star(\mathbf{u})-\mu D\le g_\mu^\star(\mathbf{u})\le g^\star(\mathbf{u}),\quad\forall\,\boldsymbol{u}\in\mathbb{R}^n. \tag{35}$$

Note that the derivation of (35) does not require g be convex, but the strong convexity of $g+\mu d$ (required by Lemma 9) relies on the convexity of g. By (35), to find an ε accurate solution to $J(\mathbf{w})$, it suffices to set the maximum deviation $\mu D<\frac{\varepsilon}{2}$ (i.e., $\mu<\frac{\varepsilon}{2D}$), and then find a $\frac{\varepsilon}{2}$ accurate solution to J_μ in (6). Initialize $\mathbf{w}$ to $\mathbf{0}$ and apply Nesterov's accelerated gradient method in Nesterov (2007) to J_μ, this takes at most

$$k=\min\left\{\sqrt{\frac{4L_\mu\Delta_0}{\varepsilon}},\log\frac{L_\mu\Delta_0}{\varepsilon}\Big/\log\left(1-\sqrt{\lambda/L_\mu}\right)\right\}$$

number of steps where $\Delta_0=\frac{1}{2}\|\mathbf{w}^*\|^2$ and $\mathbf{w}^*$ is the minimizer of $J(\mathbf{w})$. Each step involves one gradient query of $g_\mu^\star(A^\top\mathbf{w})$ and some cheap updates. Plugging in $L_\mu\le\frac{2D}{\varepsilon}\|A\|^2$ and using $\log(1+\delta)\approx\delta$ when $\delta\approx0$, we get the iteration bound in (7).

References

Alekh Agarwal, Peter Bartlett, Pradeep Ravikumar, and Martin Wainwright. Information-theoretic lower bounds on the oracle complexity of convex optimization. In *Neural Information Processing Systems*, 2009.

Satish Balay, Jed Brown, Kris Buschelman, William Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2011. http://www.mcs.anl.gov/petsc.

Peter Bartlett, Michael Jordan, and Jon McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Steve Benson, Lois Curfman McInnes, Jorge Moré, Todd Munson, and Jason Sarich. TAO user manual (revision 1.10.1). Technical Report ANL/MCS-TM-242, Mathematics and Computer Science Division, Argonne National Laboratory, 2010. http://www.mcs.anl.gov/tao.

Léon Bottou. Stochastic gradient SVMs. `http://leon.bottou.org/projects/sgd`, 2008.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, England, 2004.

Olivier Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5): 1155–1178, 2007.

Chuong Do, Quoc Le, and Chuan-Sheng Foo. Proximal regularization for online and batch learning. In *International Conference on Machine Learning ICML*, 2009.

Michael Ferris and Todd Munson. Interior-point methods for massive support vector machines. *SIAM Journal on Optimization*, 13(3):783–804, 2002.

Vojtěch Franc and Sören Sonnenburg. Optimized cutting plane algorithm for support vector machines. In Andrew McCallum and Sam Roweis, editors, *ICML*, pages 320–327. Omnipress, 2008.

Andrew Frank and Arthur Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex Analysis and Minimization Algorithms, I and II*, volume 305 and 306. Springer-Verlag, 1996.

Cho Jui Hsieh, Kai Wei Chang, Chih Jen Lin, Sathiya Keerthi, and Sundararajan Sellamanickam. A dual coordinate descent method for large-scale linear SVM. In William Cohen, Andrew McCallum, and Sam Roweis, editors, *ICML*, pages 408–415. ACM, 2008.

Thorsten Joachims. A support vector method for multivariate performance measures. In *Proc. Intl. Conf. Machine Learning*, pages 377–384, San Francisco, California, 2005. Morgan Kaufmann Publishers.

Thorsten Joachims. Training linear SVMs in linear time. In *Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD)*. ACM, 2006.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

Nikolas List and Hans Ulrich Simon. Svm-optimization and steepest-descent line search. In Sanjoy Dasgupta and Adam Klivans, editors, *Proc. Annual Conf. Computational Learning Theory*, LNCS. Springer, 2009.

Arkadi Nemirovski and David Yudin. *Problem Complexity and Method Efficiency in Optimization*. John Wiley and Sons, 1983.

Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. *Soviet Math. Docl.*, 269:543–547, 1983.

Yurii Nesterov. *Introductory Lectures On Convex Optimization: A Basic Course*. Springer, 2003.

Yurii Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, 2005.

Yurii Nesterov. Gradient methods for minimizing composite objective function. Technical Report 76, CORE Discussion Paper, UCL, 2007.

Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 2nd edition, 2006.

John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, 1998.

Bernhard Schölkopf and Alexander Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proc. Intl. Conf. Machine Learning*, 2007.

Soeren Sonnenburg, Vojtech Franc, Elad Yom-Tov, and Michele Sebag. Pascal large scale learning challenge. 2008. URL `http://largescale.ml.tu-berlin.de/workshop/`.

Sören Sonnenburg and Vojtěch Franc. COFFIN: A computational framework for linear SVMs. In *Proceedings of the International Conference on Machine Learning*, Haifa, 2010.

Choon Hui Teo, S. V. N. Vishwanthan, Alexander Smola, and Quoc Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11:311–365, January 2010.

Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Ann. Statist.*, 32(1):56–85, 2004.

Xinhua Zhang, Ankan Saha, and S. V. N. Vishwanathan. Regularized risk minimization by Nesterov's accelerated gradient methods: Algorithmic extensions and empirical studies. Technical report arXiv:1011.0472, 2010. URL `http://arxiv.org/abs/1011.0472`.

Xinhua Zhang, Ankan Saha, and S. V. N. Vishwanathan. Lower bounds on rate of convergence of cutting plane methods. In *Advances in Neural Information Processing Systems 23*, 2011a.

Xinhua Zhang, Ankan Saha, and S. V. N. Vishwanathan. Smoothing multivariate performance measures. In *Proceedings of UAI*, 2011b.

Xinhua Zhang, Ankan Saha, and S. V. N. Vishwanathan. Smoothopt, 2012. URL `http://webdocs.cs.ualberta.ca/~xinhua2/SmoothOPT`.

Tianyi Zhou, Dacheng Tao, and Xindong Wu. NESVM: a fast gradient method for support vector machines. In *Proc. Intl. Conf. Data Mining*, 2010.