Journal of Machine Learning Research 11 (2010) 311-365 Submitted 2/09; Revised 7/09; Published 1/10

Bundle Methods for Regularized Risk Minimization

Choon Hui Teo[*] CHOONHUI.TEO@ANU.EDU.AU
College of Engineering and Computer Science
Australian National University
Canberra ACT 0200, Australia

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

Alex Smola ALEX@SMOLA.ORG
Yahoo! Research
Santa Clara, CA USA

Quoc V. Le QUOCLE@STANFORD.EDU
Department of Computer Science
Stanford University
Stanford, CA 94305 USA

Editor: Thorsten Joachims

Abstract

A wide variety of machine learning problems can be described as minimizing a regularized risk functional, with different algorithms using different notions of risk and different regularizers. Examples include linear Support Vector Machines (SVMs), Gaussian Processes, Logistic Regression, Conditional Random Fields (CRFs), and Lasso amongst others. This paper describes the theory and implementation of a scalable and modular convex solver which solves all these estimation problems. It can be parallelized on a cluster of workstations, allows for data-locality, and can deal with regularizers such as L_1 and L_2 penalties. In addition to the unified framework we present tight convergence bounds, which show that our algorithm converges in $O(1/\epsilon)$ steps to ϵ precision for general convex problems and in $O(\log(1/\epsilon))$ steps for continuously differentiable problems. We demonstrate the performance of our general purpose solver on a variety of publicly available data sets.

Keywords: optimization, subgradient methods, cutting plane method, bundle methods, regularized risk minimization, parallel optimization

[*]. Also at Canberra Research Laboratory, NICTA.

1. Introduction

At the heart of many machine learning algorithms is the problem of minimizing a regularized risk functional. That is, one would like to solve

$$\min_{w} J(w) := \lambda\Omega(w) + R_{\text{emp}}(w), \tag{1}$$

$$\text{where } R_{\text{emp}}(w) := \frac{1}{m}\sum_{i=1}^{m} l(x_i, y_i, w) \tag{2}$$

is the empirical risk. Moreover, $x_i \in \mathcal{X} \subseteq \mathbb{R}^d$ are referred to as training instances and $y_i \in \mathcal{Y}$ are the corresponding labels. l is a (surrogate) convex loss function measuring the discrepancy between y and the predictions arising from using w. For instance, w might enter our model via $l(x,y,w) = (\langle w,x\rangle - y)^2$, where $\langle\cdot,\cdot\rangle$ denotes the standard Euclidean dot product. Finally, $\Omega(w)$ is a convex function serving the role of a regularizer with regularization constant $\lambda > 0$. Typically Ω is differentiable and cheap to compute. In contrast, the empirical risk term $R_{\text{emp}}(w)$ is often non-differentiable, and almost always computationally expensive to deal with.

For instance, if we consider the problem of predicting binary valued labels $y \in \{\pm 1\}$, we can set $\Omega(w) = \frac{1}{2}\|w\|_2^2$ (i.e., L_2 regularization), and the loss $l(x,y,w)$ to be the binary hinge loss, $\max(0, 1 - y\langle w,x\rangle)$, thus recovering linear Support Vector Machines (SVMs) (Joachims, 2006). Using the same regularizer but changing the loss function to $l(x,y,w) = \log(1 + \exp(-y\langle w,x\rangle))$, yields logistic regression. Extensions of these loss functions allow us to handle structure in the output space (Bakir et al., 2007) (also see Appendix A for a comprehensive exposition of many common loss functions). On the other hand, changing the regularizer $\Omega(w)$ to the sparsity inducing $\|w\|_1$ (i.e., L_1 regularization) leads to Lasso-type estimation algorithms (Mangasarian, 1965; Tibshirani, 1996; Candes and Tao, 2005).

If the objective function J is differentiable, for instance in the case of logistic regression, we can use smooth optimization techniques such as the standard quasi-Newtons methods like BFGS or its limited memory variant LBFGS (Nocedal and Wright, 1999). These methods are effective and efficient even when m and d are large (Sha and Pereira, 2003; Minka, 2007). However, it is not straightforward to extend these algorithms to optimize a non-differentiable objective, for instance, when dealing with the binary hinge loss (see, e.g., Yu et al., 2008).

When J is non-differentiable, one can use nonsmooth convex optimization techniques such as the cutting plane method (Kelly, 1960) or its *stabilized* version the bundle method (Hiriart-Urruty and Lemaréchal, 1993). The bundle methods not only stabilize the optimization procedure but make the problem a well-posed one, that is, with unique solution. However, the amount of *external* stabilization that needs to be added is a parameter that requires careful tuning.

In this paper, we bypass this stabilization parameter tuning problem by taking a different route. The resultant algorithm – Bundle Method for Regularized Risk Minimization (BMRM) – has certain desirable properties: a) it has no parameters to tune, and b) it is applicable to a wide variety of regularized risk minimization problems. Furthermore, we show that BMRM has an $O(1/\varepsilon)$ rate of convergence for nonsmooth problems and $O(\log(1/\varepsilon))$ for smooth problems. This is significantly tighter than the $O(1/\varepsilon^2)$ rates provable for standard bundle methods (Lemaréchal et al., 1995). A related optimizer, SVM$^{\text{struct}}$ (Tsochantaridis et al., 2005), which is widely used in machine learning applications was also shown to converge at $O(1/\varepsilon^2)$ rates. Our analysis also applies to SVM$^{\text{struct}}$, which we show to be a special case of our solver, and hence tightens its convergence rate to $O(1/\varepsilon)$.

Very briefly, we highlight the two major advantages of our implementation. First, it is completely modular; new loss functions, regularizers, and solvers can be added with relative ease. Second, our architecture allows the empirical risk computation (2) to be easily parallelized. This makes our solver amenable to large data sets which cannot fit into the memory of a single computer. Our open source C/C++ implementation is freely available for download.[1]

The outline of our paper is as follows. In Section 2 we describe BMRM and contrast it with standard bundle methods. We also prove rates of convergence. In Section 3 we discuss implementation issues and present principled techniques to control memory usage, as well as to speed up computation via parallelization. Section 4 puts our work in perspective, and discusses related work. Section 5 is devoted to extensive experimental evaluation, which shows that our implementation is comparable to or better than specialized state-of-the-art solvers on a number of publicly available data sets. Finally, we conclude our work and discuss related issues in Section 6. In Appendix A we describe various *classes* of loss functions organized according to their common traits in computation. Long proofs are relegated to Appendix B. Before we proceed a brief note about our notation:

1.1 Notation

The indices of elements of a sequence or a set appear in subscript, for example, u_1, u_2. The i-th component of a vector u is denoted by $u^{(i)}$. $[k]$ is the shorthand for the set $\{1, 2, \ldots, k\}$. The L_p norm is defined as $\|u\|_p = (\sum_{i=1}^d |u^{(i)}|^p)^{1/p}$, for $p \geq 1$, and we use $\|\cdot\|$ to denote $\|\cdot\|_2$ whenever the context is clear. $\mathbf{1}_d$ and $\mathbf{0}_d$ denote the d-dimensional vectors of all ones and zeros respectively.

2. Bundle Methods

The precursor to the bundle methods is the cutting plane method (CPM) (Kelly, 1960). CPM uses subgradients, which are a generalization of gradients appropriate for convex functions, including those which are not necessarily smooth. Suppose w' is a point where a convex function J is finite. Then a subgradient is the normal vector of any tangential supporting hyperplane of J at w' (see Figure 1 for geometric intuition). Formally s' is called a subgradient of J at w' if, and only if,

$$J(w) \geq J(w') + \langle w - w', s' \rangle \quad \forall w. \tag{3}$$

The set of all subgradients at w' is called the subdifferential, and is denoted by $\partial_w J(w')$. If this set is not empty then J is said to be *subdifferentiable at* w'. On the other hand, if this set is a singleton then the function is said to be *differentiable* at w'. Convex functions are subdifferentiable everywhere in their domain (Hiriart-Urruty and Lemaréchal, 1993).

As implied by (3), J is bounded from below by its linearization (i.e., first order Taylor approximation) at w'. Given subgradients $s_1, s_2, \ldots, s_t$ evaluated at locations $w_0, w_1, \ldots, w_{t-1}$, we can state a tighter (piecewise linear) lower bound for J as follows

$$J(w) \geq J_t^{\mathrm{CP}}(w) := \max_{1 \leq i \leq t} \{J(w_{i-1}) + \langle w - w_{i-1}, s_i \rangle\}. \tag{4}$$

This lower bound forms the basis of the CPM, where at iteration t the set $\{w_i\}_{i=0}^{t-1}$ is augmented by

$$w_t := \operatorname*{argmin}_w J_t^{\mathrm{CP}}(w).$$

1. Software available at `http://users.rsise.anu.edu.au/~chteo/BMRM.html`.

This iteratively refines the piecewise linear lower bound J^{CP} and allows us to get close to the minimum of J (see Figure 2 for an illustration).

If w^* denotes the minimizer of J, then clearly each $J(w_i) \geq J(w^*)$ and hence $\min_{0 \leq i \leq t} J(w_i) \geq J(w^*)$. On the other hand, since $J \geq J_t^{\mathrm{CP}}$ it follows that $J(w^*) \geq J_t^{\mathrm{CP}}(w_t)$. In other words, $J(w^*)$ is sandwiched between $\min_{0 \leq i \leq t} J(w_i)$ and $J_t^{\mathrm{CP}}(w_t)$ (see Figure 3 for an illustration). The CPM monitors the monotonically decreasing quantity

$$\varepsilon_t := \min_{0 \leq i \leq t} J(w_i) - J_t^{\mathrm{CP}}(w_t),$$

and terminates whenever ε_t falls below a predefined threshold ε. This ensures that the solution $J(w_t)$ satisfies $J(w_t) \leq J(w^*) + \varepsilon$.

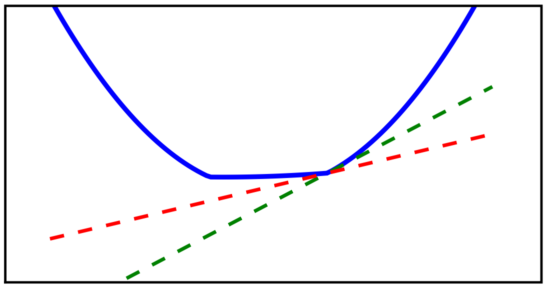


Figure 1: Geometric intuition of a subgradient. The nonsmooth convex function (solid blue) is only subdifferentiable at the "kink" points. We illustrate two of its subgradients (dashed green and red lines) at a "kink" point which are tangential to the function. The normal vectors to these lines are subgradients.

2.1 Standard Bundle Methods

Although CPM was shown to be convergent (Kelly, 1960), it is well known (see, e.g., Lemaréchal et al., 1995; Belloni, 2005) that CPM can be very slow when new iterates move too far away from the previous ones (i.e., causing unstable "zig-zag" behavior in the iterates).

Bundle methods stabilize CPM by augmenting the piecewise linear lower bound (e.g., $J_t^{\mathrm{CP}}(w)$ as in (4)) with a prox-function (i.e., proximity control function) which prevents overly large steps in the iterates (Kiwiel, 1990). Roughly speaking, there are 3 popular types of bundle methods, namely, *proximal* (Kiwiel, 1990), *trust region* (Schramm and Zowe, 1992), and *level set* (Lemaréchal et al., 1995).[2] All three versions use $\frac{1}{2}\|\cdot\|^2$ as their prox-function, but differ in the way they compute the

2. For brevity we will only describe "first-order" bundle methods, and omit discussion about "second-order" variants such as the bundle-Newton method of Lukšan and Vlček (1998).

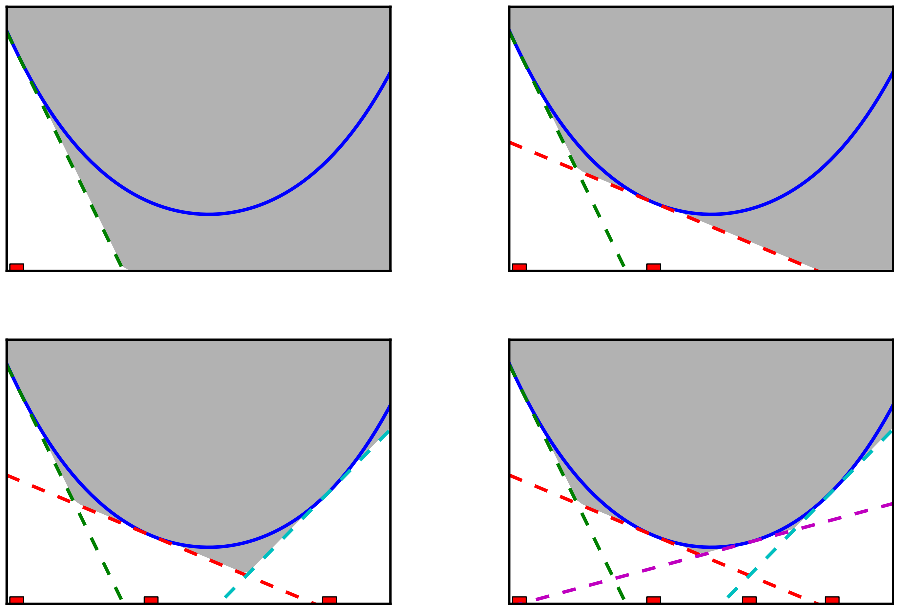


Figure 2: A convex function (blue solid curve) is bounded from below by its linearizations (dashed lines). The gray area indicates the piecewise linear lower bound obtained by using the linearizations. We depict a few iterations of the cutting plane method. At each iteration the piecewise linear lower bound is minimized and a new linearization is added at the minimizer (red rectangle). As can be seen, adding more linearizations improves the lower bound.

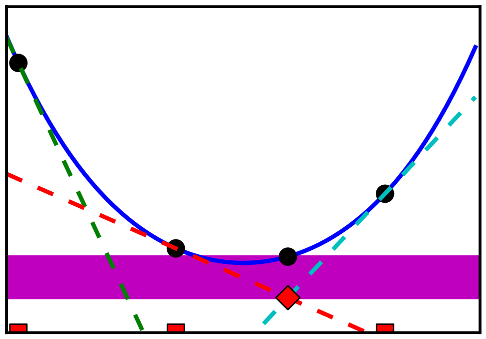


Figure 3: A convex function (blue solid curve) with three linearizations (dashed lines) evaluated at three different locations (red squares). The approximation gap ϵ_3 at the end of third iteration is indicated by the height of the magenta horizontal band, that is, difference between lowest value of $J(w)$ evaluated so far (lowest black circle) and the minimum of $J_3^{\mathrm{CP}}(w)$ (red diamond).

new iterate:

$$\text{proximal:} \quad w_t := \operatorname*{argmin}_w \{ \tfrac{\zeta_t}{2} \|w - \hat{w}_{t-1}\|^2 + J_t^{\text{CP}}(w) \}, \tag{5}$$

$$\text{trust region:} \quad w_t := \operatorname*{argmin}_w \{ J_t^{\text{CP}}(w) \mid \tfrac{1}{2} \|w - \hat{w}_{t-1}\|^2 \leq \kappa_t \}, \tag{6}$$

$$\text{level set:} \quad w_t := \operatorname*{argmin}_w \{ \tfrac{1}{2} \|w - \hat{w}_{t-1}\|^2 \mid J_t^{\text{CP}}(w) \leq \tau_t \},$$

where $\hat{w}_{t-1}$ is the current prox-center, and ζ_t, κ_t, and τ_t are positive trade-off parameters of the stabilization. Although (5) can be shown to be equivalent to (6) for appropriately chosen ζ_t and κ_t, tuning ζ_t is rather difficult while a trust region approach can be used for automatically tuning κ_t. Consequently the trust region algorithm BT of Schramm and Zowe (1992) is widely used in practice.

Since our methods (see Section 2.2) are closely related to the proximal bundle method, we will now describe them in detail. Similar to the CPM the proximal bundle method also builds a piecewise linear lower bound J_t^{CP} (see (4)). In contrast to the CPM, the piecewise linear lower bound augmented with a stabilization term $\frac{\zeta_t}{2} \|w - \hat{w}_{t-1}\|^2$, is minimized to produce the intermediate iterate $\bar{w}_t$. The approximation gap in this case includes the prox-function:

$$\varepsilon_t := J(\hat{w}_{t-1}) - \left[J_t^{\text{CP}}(\bar{w}_t) + \frac{\zeta_t}{2} \|\bar{w}_t - \hat{w}_{t-1}\|^2 \right].$$

If ε_t is less than the pre-defined threshold ε the algorithm exits. Otherwise, a line search is performed along the line joining $\hat{w}_{t-1}$ and $\bar{w}_t$ to produce the new iterate w_t. If w_t results in a sufficient decrease of the objective function then it is accepted as the new prox-center $\hat{w}_t$; this is called a serious step. Otherwise, the prox-center remains the same; this is called a null step. Detailed pseudocode can be found in Algorithm 1.

If the approximation gap ε_t is smaller than ε, then this ensures that the solution $J(\hat{w}_{t-1})$ satisfies $J(\hat{w}_{t-1}) \leq J(w) + \frac{\zeta_t}{2} \|w - \hat{w}_{t-1}\|^2 + \varepsilon$ for all w. In particular, if $J(w^*)$ denotes the optimum as before, then $J(\hat{w}_{t-1}) \leq J(w^*) + \frac{\zeta_t}{2} \|w^* - \hat{w}_{t-1}\|^2 + \varepsilon$. Contrast this with the approximation guarantee of the CPM, which does not involve the $\frac{\zeta_t}{2} \|w^* - \hat{w}_{t-1}\|^2$ term.

Although the positive coefficient ζ_t is assumed fixed throughout the algorithm, in practice it must be updated after every iteration to achieve faster convergence, and to guarantee a good quality solution (Kiwiel, 1990). Same is the case for κ_t and τ_t in trust region and level set bundle methods, respectively. Although the update is not difficult, the procedure relies on other parameters which require careful *tuning* (Kiwiel, 1990; Schramm and Zowe, 1992; Lemaréchal et al., 1995).

In the next section, we will describe our method (BMRM) which avoids this problem. There are two key differences between BMRM and the proximal bundle method: Firstly, BMRM maintains a piecewise linear lower bound of $R_{\text{emp}}(w)$ instead of $J(w)$. Secondly, the the stabilizer (i.e., $\|w - \hat{w}_t\|^2$) in proximal bundle method is replaced by the regularizer $\Omega(w)$ hence there is no stabilization parameter to tune. As we will see, not only is the implementation straightforward, but the rates of convergence also improve from $O(1/\varepsilon^3)$ or $O(1/\varepsilon^2)$ to $O(1/\varepsilon)$.

Algorithm 1 Proximal Bundle Method

```
1:  input & initialization: $\epsilon \geq 0$, $\rho \in (0,1)$, $w_0$, $t \leftarrow 0$, $\hat{w}_0 \leftarrow w_0$
2:  loop
3:      $t \leftarrow t+1$
4:      Compute $J(w_{t-1})$ and $s_t \in \partial_w J(w_{t-1})$
5:      Update model $J_t^{\text{CP}}(w) := \max_{1\leq i\leq t}\{J(w_{i-1}) + \langle w - w_{i-1}, s_i\rangle\}$
6:      $\bar{w}_t \leftarrow \text{argmin}_w J_t^{\text{CP}}(w) + \frac{\zeta_t}{2}\|w - \hat{w}_{t-1}\|^2$
7:      $\epsilon_t \leftarrow J(\hat{w}_{t-1}) - \left[J_t^{\text{CP}}(\bar{w}_t) + \frac{\zeta_t}{2}\|\bar{w}_t - \hat{w}_{t-1}\|^2\right]$
8:      if $\epsilon_t < \epsilon$ then return $\bar{w}_t$
9:      Linesearch: $\eta_t \leftarrow \text{argmin}_{\eta\in\mathbb{R}} J(\hat{w}_{t-1} + \eta(\bar{w}_t - \hat{w}_{t-1}))$        (if expensive, set $\eta_t = 1$)
10:     $w_t \leftarrow \hat{w}_{t-1} + \eta_t(\bar{w}_t - \hat{w}_{t-1})$
11:     if $J(\hat{w}_{t-1}) - J(w_t) \geq \rho\epsilon_t$ then
12:         SERIOUS STEP: $\hat{w}_t \leftarrow w_t$
13:     else
14:         NULL STEP: $\hat{w}_t \leftarrow \hat{w}_{t-1}$
15:     end if
16: end loop
```

2.2 Bundle Methods for Regularized Risk Minimization (BMRM)

Define:

$$\begin{aligned}
&\text{(subgradient of } R_{\text{emp}}) && a_t \in \partial_w R_{\text{emp}}(w_{t-1}),\\
&\text{(offset)} && b_t := R_{\text{emp}}(w_{t-1}) - \langle w_{t-1}, a_t\rangle,\\
&\text{(piecewise linear lower bound of } R_{\text{emp}}) && R_t^{\text{CP}}(w) := \max_{1\leq i\leq t}\{\langle w, a_i\rangle + b_i\},\\
&\text{(piecewise convex lower bound of } J) && J_t(w) := \lambda\Omega(w) + R_t^{\text{CP}}(w),\\
&\text{(iterate)} && w_t := \min_w J_t(w),\\
&\text{(approximation gap)} && \epsilon_t := \min_{0\leq i\leq t} J(w_i) - J_t(w_t).
\end{aligned}$$

We now describe BMRM (Algorithm 2), and contrast it with the proximal bundle method. At iteration t the algorithm builds the lower bound R_t^{CP} to the empirical risk R_{emp}. The new iterate w_t is then produced by minimizing J_t which is R_t^{CP} augmented with the regularizer Ω; this is the key difference from the proximal bundle method which uses the $\frac{\zeta_t}{2}\|w - \hat{w}_{t-1}\|^2$ prox-function for stabilization. The algorithm repeats until the approximation gap ϵ_t is less than the pre-defined threshold ϵ. Unlike standard bundle methods there is no notion of a serious or null step in our algorithm. In fact, our algorithm does not even maintain a prox-center. It can be viewed as a special case of standard bundle methods where the prox-center is always the origin and never updated (hence every step is a null step). Furthermore, unlike the proximal bundle method, the approximation guarantees of our algorithm do not involve the $\frac{\zeta_t}{2}\|w^* - w_t\|^2$ term.

Algorithm 2 is simple and easy to implement as it does not involve a line search. In fact, whenever efficient (exact) line search is available, it can be used to achieve faster convergence as

Algorithm 2 BMRM

```
1: input & initialization: $\varepsilon \geq 0$, $w_0$, $t \leftarrow 0$
2: repeat
3:     $t \leftarrow t+1$
4:     Compute $a_t \in \partial_w R_{\text{emp}}(w_{t-1})$ and $b_t \leftarrow R_{\text{emp}}(w_{t-1}) - \langle w_{t-1}, a_t \rangle$
5:     Update model: $R_t^{\text{CP}}(w) := \max_{1 \leq i \leq t} \{\langle w, a_i \rangle + b_i\}$
6:     $w_t \leftarrow \operatorname{argmin}_w J_t(w) := \lambda\Omega(w) + R_t^{\text{CP}}(w)$
7:     $\varepsilon_t \leftarrow \min_{0 \leq i \leq t} J(w_i) - J_t(w_t)$
8: until $\varepsilon_t \leq \varepsilon$
9: return $w_t$
```

observed by Franc and Sonnenburg (2008) in the case of linear SVMs with binary hinge loss.[3] We now turn to a variant of BMRM which uses a line search (Algorithm 3); this is a generalization of the optimized cutting plane algorithm for support vector machines (OCAS) of Franc and Sonnenburg (2008). This variant first builds R_t^{CP} and minimizes J_t to obtain an intermediate iterate w_t. Then, it performs a line search along the line joining w_{t-1}^b and w_t to produce w_t^b which acts like the new prox-center. Note that $w_t - w_{t-1}^b$ is not necessarily a direction of descent; therefore the line search might return a zero step. Instead of using w_t^b as the new iterate the algorithm uses the pre-set parameter θ to generate w_t^c on the line segment joining w_t^b and w_t. Franc and Sonnenburg (2008) report that setting $\theta = 0.9$ works well in practice. It is easy to see that Algorithm 3 reduces to Algorithm 2 if we set $\eta_t = 1$ for all t, and use the same termination criterion. It is worthwhile noting that this variant is not applicable for structured learning problems such as Max-Margin Markov Networks (Taskar et al., 2004), because no efficient line search is known for such problems.

A specialized variant of BMRM which handles quadratic regularizers, that is, $\Omega(w) = \frac{1}{2}\|w\|^2$ was first introduced to the machine learning community by Tsochantaridis et al. (2005) as SVM$^{\text{struct}}$. In particular, SVM$^{\text{struct}}$ handles quadratic regularizers $\Omega(w) = \frac{1}{2}\|w\|^2$ and non-differentiable large margin loss functions such as (24). Its 1-slack formulation (Joachims et al., 2009) can be shown to be equivalent to BMRM for this specific type of regularizer and loss function. Somewhat confusingly, these algorithms are called the cutting plane method even though they are closer in spirit to bundle methods.

2.3 Dual Problems

In this section, we describe how the sub-problem

$$w_t = \operatorname*{argmin}_w J_t(w) := \lambda\Omega(w) + \max_{1 \leq i \leq t} \langle w, a_i \rangle + b_i \tag{7}$$

in Algorithms 2 and 3 is solved via a dual formulation. In fact, we will show that we need not know $\Omega(w)$ at all, instead it is sufficient to work with its Fenchel dual (Hiriart-Urruty and Lemaréchal, 1993):

Definition 1 (Fenchel Dual) *Denote by $\Omega : \mathcal{W} \to \mathbb{R}$ a convex function on a convex set $\mathcal{W}$. Then the dual Ω^* of Ω is defined as*

$$\Omega^*(\mu) := \sup_{w \in \mathcal{W}} \langle w, \mu \rangle - \Omega(w). \tag{8}$$

3. A different optimization method but with identical efficient line search procedure is described in Yu et al. (2008).

Algorithm 3 BMRM with Line Search

```
1: input & initialization: $\epsilon \geq 0$, $\theta \in (0,1]$, $w_0^b$, $w_0^c \leftarrow w_0^b$, $t \leftarrow 0$
2: repeat
3:    $t \leftarrow t+1$
4:    Compute $a_t \in \partial_w R_{\text{emp}}(w_{t-1}^c)$, and $b_t \leftarrow R_{\text{emp}}(w_{t-1}^c) - \langle w_{t-1}^c, a_t \rangle$
5:    Update model: $R_t^{\text{CP}}(w) := \max_{1 \leq i \leq t}\{\langle w, a_i \rangle + b_i\}$
6:    $w_t \leftarrow \operatorname{argmin}_w J_t(w) := \lambda\Omega(w) + R_t^{\text{CP}}(w)$
7:    Linesearch: $\eta_t \leftarrow \operatorname{argmin}_{\eta \in \mathbb{R}} J(w_{t-1}^b + \eta(w_t - w_{t-1}^b))$
8:    $w_t^b \leftarrow w_{t-1}^b + \eta_t(w_t - w_{t-1}^b)$
9:    $w_t^c \leftarrow (1-\theta)w_t^b + \theta w_t$
10:   $\epsilon_t \leftarrow J(w_t^b) - J_t(w_t)$
11: until $\epsilon_t \leq \epsilon$
12: return $w_t^b$
```

Several choices of regularizers are common. For $\mathcal{W} = \mathbb{R}^d$ the squared norm regularizer yields

$$\Omega(w) = \frac{1}{2}\|w\|_2^2 \qquad \text{and} \qquad \Omega^*(\mu) = \frac{1}{2}\|\mu\|_2^2.$$

More generally, for L_p norms one obtains (Boyd and Vandenberghe, 2004; Shalev-Shwartz and Singer, 2006):

$$\Omega(w) = \frac{1}{2}\|w\|_p^2 \qquad \text{and} \qquad \Omega^*(\mu) = \frac{1}{2}\|\mu\|_q^2 \text{ where } \frac{1}{p} + \frac{1}{q} = 1.$$

For any positive definite matrix B, we can construct a quadratic form regularizer which allows non-uniform penalization of the weight vector as:

$$\Omega(w) = \frac{1}{2}w^\top Bw \qquad \text{and} \qquad \Omega^*(\mu) = \frac{1}{2}\mu^\top B^{-1}\mu.$$

For the *unnormalized* negative entropy, where $\mathcal{W} = \mathbb{R}_+^d$, we have

$$\Omega(w) = \sum_i w^{(i)} \log w^{(i)} \qquad \text{and} \qquad \Omega^*(\mu) = \sum_i \exp \mu^{(i)}.$$

For the *normalized* negative entropy, where $\mathcal{W} = \{w \mid w \geq 0 \text{ and } \|w\|_1 = 1\}$ is the probability simplex, we have

$$\Omega(w) = \sum_i w^{(i)} \log w^{(i)} \qquad \text{and} \qquad \Omega^*(\mu) = \log \sum_i \exp \mu^{(i)}.$$

If Ω is differentiable the w at which the supremum of (8) is attained can be written as $w = \partial_\mu \Omega^*(\mu)$ (Boyd and Vandenberghe, 2004). In the sequel we will always assume that Ω^* is twice differentiable. Note that all the regularizers we discussed above are twice differentiable. The following theorem states the dual problem of (7).

Theorem 2 *Denote by $A = [a_1, \ldots, a_t]$ the matrix whose columns are the (sub)gradients, and let $b = [b_1, \ldots, b_t]$. The dual problem of*

$$w_t = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \{J_t(w) := \max_{1 \le i \le t} \langle w, a_i \rangle + b_i + \lambda\Omega(w)\} \quad \textit{is} \tag{9}$$

$$\alpha_t = \underset{\alpha \in \mathbb{R}^t}{\operatorname{argmax}} \{J_t^*(\alpha) := -\lambda\Omega^*(-\lambda^{-1}A\alpha) + \alpha^\top b \mid \alpha \ge 0, \ \|\alpha\|_1 = 1\}. \tag{10}$$

Furthermore, w_t and α_t are related by the dual connection $w_t = \partial\Omega^*(-\lambda^{-1}A\alpha_t)$.

Proof We rewrite (9) as a constrained optimization problem: $\min_{w,\xi} \lambda\Omega(w) + \xi$ subject to $\xi \ge \langle w, a_i \rangle + b_i$ for $i = 1, \ldots, t$. By introducing non-negative Lagrange multipliers α and recalling that $\mathbf{1}_t$ denotes the t dimensional vector of all ones, the corresponding Lagrangian can be written as

$$L(w, \xi, \alpha) = \lambda\Omega(w) + \xi - \alpha^\top \left(\xi\mathbf{1}_t - A^\top w - b\right) \text{ with } \alpha \ge 0, \tag{11}$$

where $\alpha \ge 0$ denotes that each component of α is non-negative. Taking derivatives with respect to ξ yields $1 - \alpha^\top \mathbf{1}_t = 0$. Moreover, minimization of L with respect to w implies solving $\max_w \langle w, -\lambda^{-1}A\alpha \rangle - \Omega(w) = \Omega^*(-\lambda^{-1}A\alpha)$. Plugging both terms back into (11) we eliminate the primal variables ξ and w. ■

Since Ω^* is assumed to be twice differentiable and the constraints of (10) are simple, one can easily solve (10) with standard smooth optimization methods such as the *penalty/barrier* methods (Nocedal and Wright, 1999). Recall that for the square norm regularizer $\Omega(w) = \frac{1}{2}\|w\|_2^2$, commonly used in SVMs and Gaussian Processes, the Fenchel dual is given by $\Omega^*(\mu) = \frac{1}{2}\|\mu\|_2^2$. The following corollary is immediate:

Corollary 3 *For quadratic regularization, that is, $\Omega(w) = \frac{1}{2}\|w\|_2^2$,* (10) *becomes*

$$\alpha_t = \underset{\alpha \in \mathbb{R}^t}{\operatorname{argmax}} \{-\tfrac{1}{2\lambda}\alpha^\top A^\top A\alpha + \alpha^\top b \mid \alpha \ge 0, \ \|\alpha\|_1 = 1\}.$$

This means that for quadratic regularization the dual optimization problem is a quadratic program (QP) where the number of constraints equals the number of (sub)gradients computed previously. Since t is typically in the order of 10s to 100s, the resulting QP is very cheap to solve. In fact, we do not even need to know the (sub)gradients explicitly. All that is required to define the QP are the inner products between (sub)gradients $\langle a_i, a_j \rangle$.

2.4 Convergence Analysis

While the variants of bundle methods we proposed are intuitively plausible, it remains to be shown that they have good rates of convergence. In fact, past results, such as those by Tsochantaridis et al. (2005) suggest a slow $O(1/\varepsilon^2)$ rate of convergence. In this section we tighten their results and show an $O(1/\varepsilon)$ rate of convergence for nonsmooth loss functions and $O(\log(1/\varepsilon))$ rates for smooth loss functions under mild assumptions. More concretely we prove the following two convergence results:

(a) Assume that $\max_{u \in \partial_w R_{\text{emp}}(w)} \|u\| \le G$. For regularizers $\Omega(w)$ for which $\left\|\partial_\mu^2 \Omega^*(\mu)\right\| \le H^*$ we prove $O(1/\varepsilon)$ rate of convergence, that is, we show that our algorithm converges to within ε of the optimal solution in $O(1/\varepsilon)$ iterations.

(b) Under the above conditions, if furthermore $\left\|\partial_w^2 J(w)\right\| \leq H$, that is, the Hessian of J is bounded, we can show $O(\log(1/\varepsilon))$ rate of convergence.

For our convergence proofs we use a duality argument similar to those put forward in Shalev-Shwartz and Singer (2006) and Tsochantaridis et al. (2005), both of which share key techniques with Zhang (2003). Recall that ε_t denotes our approximation gap, which in turn upper bounds how far away we are from the optimal solution. In other words, $\varepsilon_t \geq \min_{0 \leq i \leq t} J(w_i) - J^*$, where J^* denotes the optimum value of the objective function J. The quantity $\varepsilon_t - \varepsilon_{t+1}$ can thus be viewed as the "progress" made towards J^* in iteration t. The crux of our proof argument lies in showing that for nonsmooth loss functions the recurrence $\varepsilon_t - \varepsilon_{t+1} \geq c \cdot \varepsilon_t^2$ holds for some appropriately chosen constant c. The rates follow by invoking a lemma from Abe et al. (2001). In the case of the smooth losses we show that $\varepsilon_t - \varepsilon_{t+1} \geq c' \cdot \varepsilon_t$ thus implying an $O(\log(1/\varepsilon))$ rate of convergence.

In order to show the required recurrence, we first observe that by strong duality the values of the primal and dual problems (9) and (10) are equal at optimality. Hence, any progress in J_{t+1} can be computed in the dual. Next, we observe that the solution of the dual problem (10) at iteration t, denoted by α_t, forms a feasible set of parameters for the dual problem (10) at iteration $t+1$ by means of the parameterization $(\alpha_t, 0)$, that is, by padding α_t with a 0. The value of the objective function in this case equals $J_t(w_t)$.

To obtain a lower bound on the improvement due to $J_{t+1}(w_{t+1})$ we perform a 1-d optimization along $((1-\eta)\alpha_t, \eta)$ in (10). The constraint $\eta \in (0,1)$ ensures dual feasibility. We will then bound this improvement in terms of ε_t. Note that, in general, solving the dual problem (10) results in a increase which is larger than that obtained via the line search. The 1-d minimization is used only for analytic tractability. We now state our key theorem and prove it in Appendix B.

Theorem 4 *Assume that* $\max_{u \in \partial_w R_{\text{emp}}(w)} \|u\| \leq G$ *for all* $w \in \operatorname{dom} J$. *Also assume that* Ω^* *has bounded curvature, that is,* $\left\|\partial_\mu^2 \Omega^*(\mu)\right\| \leq H^*$ *for all* $\mu \in \{-\lambda^{-1} \sum_{i=1}^{t+1} \alpha_i a_i$ *where* $\alpha_i \geq 0, \ \forall i$ *and* $\sum_{i=1}^{t+1} \alpha_i = 1\}$. *In this case we have*

$$\varepsilon_t - \varepsilon_{t+1} \geq \tfrac{\varepsilon_t}{2} \min(1, \lambda \varepsilon_t / 4G^2 H^*). \tag{12}$$

Furthermore, if $\left\|\partial_w^2 J(w)\right\| \leq H$, *then we have*

$$\varepsilon_t - \varepsilon_{t+1} \geq \begin{cases} \varepsilon_t/2 & \textit{if } \varepsilon_t > 4G^2 H^*/\lambda \\ \lambda/8H^* & \textit{if } 4G^2 H^*/\lambda \geq \varepsilon_t \geq H/2 \\ \lambda \varepsilon_t / 4HH^* & \textit{otherwise.} \end{cases}$$

Note that the error keeps on halving initially and settles for a somewhat slower rate of convergence after that, whenever the Hessian of the overall risk is bounded from above. The reason for the difference in the convergence bound for differentiable and non-differentiable losses is that in the former case the gradient of the risk converges to 0 as we approach optimality, whereas in the former case, no such guarantees hold (e.g., when minimizing $|x|$ the (sub)gradient does not vanish at the optimum). The dual of many regularizers, for example, norm, squared L_p norm, and the entropic regularizer have bounded second derivative. See, for example, Shalev-Shwartz and Singer (2006) for a discussion and details. Thus our condition $\left\|\partial_\mu^2 \Omega^*(\mu)\right\| \leq H^*$ is not unreasonable. We are now in a position to state our convergence results. The proof is in Appendix B.

Theorem 5 *Assume that $J(w) \geq 0$ for all w. Under the assumptions of Theorem 4 we can give the following convergence guarantee for Algorithm 2. For any $\epsilon < 4G^2H^*/\lambda$ the algorithm converges to the desired precision after*

$$n \leq \log_2 \frac{\lambda J(0)}{G^2H^*} + \frac{8G^2H^*}{\lambda\epsilon} - 1$$

steps. Furthermore if the Hessian of $J(w)$ is bounded, convergence to any $\epsilon \leq H/2$ takes at most the following number of steps:

$$n \leq \log_2 \frac{\lambda J(0)}{4G^2H^*} + \frac{4H^*}{\lambda} \max\left[0, H - 8G^2H^*/\lambda\right] + \frac{4HH^*}{\lambda} \log(H/2\epsilon).$$

Several observations are in order: First, note that the number of iterations only depends *logarithmically* on how far the initial value $J(0)$ is away from the optimal solution. Compare this to the result of Tsochantaridis et al. (2005), where the number of iterations is linear in $J(0)$.

Second, we have an $O(1/\epsilon)$ dependence in the number of iterations in the non-differentiable case, as opposed to the $O(1/\epsilon^2)$ rates of Tsochantaridis et al. (2005). In addition to that, the convergence is $O(\log(1/\epsilon))$ for continuously differentiable problems.

Note that whenever R_{emp} is the average over many piecewise linear functions, R_{emp} behaves essentially like a function with bounded Hessian as long as we are taking large enough steps not to "notice" the fact that the term is actually nonsmooth.

Remark 6 *For $\Omega(w) = \frac{1}{2}\|w\|^2$ the dual Hessian is exactly $H^* = 1$. Moreover we know that $H \geq \lambda$ since $\left\|\partial_w^2 J(w)\right\| = \lambda + \left\|\partial_w^2 R_{\text{emp}}(w)\right\|$.*

Effectively the rate of convergence of the algorithm is governed by upper bounds on the primal and dual curvature of the objective function. This acts like a condition number of the problem—for $\Omega(w) = \frac{1}{2}w^\top Qw$ the dual is $\Omega^*(z) = \frac{1}{2}z^\top Q^{-1}z$, hence the largest eigenvalues of Q and Q^{-1} would have a significant influence on the convergence.

In terms of λ the number of iterations needed for convergence is $O(\lambda^{-1})$. In practice the iteration count *does* increase with λ, albeit not as badly as predicted. This is likely due to the fact that the empirical risk R_{emp} is typically rather smooth and has a certain inherent curvature which acts as a natural regularizer in addition to the regularization afforded by $\lambda\Omega(w)$.

For completeness we also state the convergence guarantees for Algorithm 3 and provide a proof in Appendix B.3.

Theorem 7 *Under the assumptions of Theorem 4 Algorithm 3 converges to the desired precision ϵ after*

$$n \leq \frac{8G^2H^*}{\lambda\epsilon}$$

steps for any $\epsilon < 4G^2H^*/\lambda$.

3. Implementation Issues

In this section, we discuss the memory and computational issues of the implementation of BMRM. In addition, we provide two variants of BMRM: one is memory efficient and the other one is parallelized.

3.1 Solving the BMRM Subproblem (7) with Limited Memory Space

In Section 2.3 we mentioned the dual of subproblem (7) (i.e., (10)) which is usually easier to solve when the dimensionality d of the problem is larger than the number of iterations t required by BMRM to reach desired precision ε. Although t is usually in the order of 10^2, a problem with d in the order of 10^6 or higher may use up all memory of a typical machine to store the bundle, that is, linearizations $\{(a_i, b_i)\}$, before the convergence is achieved.[4] Here we describe a principled technique which controls the memory usage while maintaining convergence guarantees.

Note that at iteration t, before the computation for new iterate w_t, Algorithm 2 maintains a bundle of t (sub)gradients $\{a_i\}_{i=1}^{t}$ of R_{emp} computed at the locations $\{w_i\}_{i=0}^{t-1}$. Furthermore, the Lagrange multipliers α_{t-1} obtained in iteration $t-1$ satisfy $\alpha_{t-1} \geq 0$ and $\sum_{i=1}^{t-1} \alpha_{t-1}^{(i)} = 1$ by the constraints of (10). We define the *aggregated* (sub)gradient $\hat{a}_I$, offset $\hat{b}_I$ and Lagrange multiplier $\hat{\alpha}_{t-1}^{(I)}$ as

$$\hat{a}_I := \frac{1}{\hat{\alpha}_{t-1}^{(I)}} \sum_{i \in I} \alpha_{t-1}^{(i)} a_i, \quad \hat{b}_I := \frac{1}{\hat{\alpha}_{t-1}^{(I)}} \sum_{i \in I} \alpha_{t-1}^{(i)} b_i, \quad \text{and} \quad \hat{\alpha}_{t-1}^{(I)} := \sum_{i \in I} \alpha_{t-1}^{(i)},$$

respectively, where $I \subseteq [t-1]$ is an index set (Kiwiel, 1983). Clearly, the optimality of (10) at the end of iteration $t-1$ is maintained when a subset $\left\{(a_i, b_i, \alpha_{t-1}^{(i)})\right\}_{i \in I}$ is replaced by the aggregate $(\hat{a}_I, \hat{b}_I, \hat{\alpha}_{t-1}^{(I)}))$ for any $I \subseteq [t-1]$.

To obtain a new iterate w_t via (10) with memory space for at most k linearizations, we can, for example, replace $\{(a_i, b_i)\}_{i \in I}$ with $(\hat{a}_I, \hat{b}_I)$ where $I = [t-k+1]$ and $2 \leq k \leq t$. Then, we solve a k-dimensional variant of (10) with $A := [\hat{a}_I, a_{t-k+2}, \ldots, a_t]$, $b := [\hat{b}_I, b_{t-k+2}, \ldots, b_t]$, and $\alpha \in \mathbb{R}^k$. The optimum of this variant will be lower than or equal to that of (10) as the latter has higher degree of freedom than the former. Nevertheless, solving this variant with $2 \leq k \leq t$ will still guarantee convergence (recall that our convergence proof only uses $k = 2$). In the sequel we name the aforementioned number k as the "bundle size" since it indicates the number of linearizations the algorithm keeps.

For concreteness, we provide here a memory efficient BMRM variant for the cases where $\Omega(w) = \frac{1}{2}\|w\|_2^2$ and $k = 2$. We first see that the dual of subproblem (7) now reads:

$$\begin{aligned} \eta &= \underset{0 \leq \eta \leq 1}{\operatorname{argmax}} -\tfrac{1}{2\lambda} \left\| \hat{a}_{[t-1]} + \eta(a_t - \hat{a}_{[t-1]}) \right\|_2^2 + \hat{b}_{[t-1]} + \eta(b_t - \hat{b}_{[t-1]}) \\ &\equiv \underset{0 \leq \eta \leq 1}{\operatorname{argmax}} -\tfrac{\eta}{\lambda} \hat{a}_{[t-1]}^{\top}(a_t - \hat{a}_{[t-1]}) - \tfrac{\eta^2}{2\lambda} \left\| a_t - \hat{a}_{[t-1]}^{\top} \right\|^2 + \eta(b_t - \hat{b}_{[t-1]}). \end{aligned} \tag{13}$$

Since (13) is quadratic in η, we can obtain the optimal η by setting the derivative of the objective in (13) to zero and clipping η in the range $[0, 1]$:

$$\eta = \min\left(\max\left(0, \frac{b_t - \hat{b}_{[t-1]} + w_{t-1}^{\top} a_t + \lambda \|w_{t-1}\|^2}{\frac{1}{\lambda}\|a_t + \lambda w_{t-1}\|^2}\right), 1\right) \tag{14}$$

4. In practice, we can remove those linearizations $\{(a_i, b_i)\}$ whose Lagrange multipliers α_i are 0 after solving (10). Although this heuristic works well and does not affect the convergence guarantee, there is no bound on the minimum number of linearizations with non-zero Lagrange multipliers needed to achieve convergence.

where $w_{t-1} = -\frac{1}{\lambda}\hat{a}_{[t-1]}$ by the dual connection. With the optimal η, we obtain the new primal iterate $w_t = (1-\eta)w_{t-1} - (\eta/\lambda)a_t$. Algorithm 4 lists the details. Note that this variant is simple to implement and does not require a QP solver.

Algorithm 4 BMRM with Aggregation of Previous Linearizations

```
 1: input & initialization: $\epsilon \geq 0$, $w_0$, $t \leftarrow 1$
 2: Compute $a_1 \in \partial_w R_{\text{emp}}(w_0)$, and $b_1 \leftarrow R_{\text{emp}}(w_0) - \langle w_0, a_1 \rangle$
 3: $w_1 \leftarrow -\frac{1}{\lambda} a_1$
 4: $\hat{b}_{[1]} \leftarrow b_1$
 5: repeat
 6:     $t \leftarrow t+1$
 7:     Compute $a_t \in \partial_w R_{\text{emp}}(w_{t-1})$ and $b_t \leftarrow R_{\text{emp}}(w_{t-1}) - \langle w_{t-1}, a_t \rangle$
 8:     Compute $\eta$ using Eq. (14)
 9:     $w_t \leftarrow (1-\eta)w_{t-1} - (\eta/\lambda)a_t$
10:     $\hat{b}_{[t]} \leftarrow (1-\eta)\hat{b}_{[t-1]} + \eta b_t$
11:     $\epsilon_t \leftarrow \min_{0 \leq i \leq t} \frac{\lambda}{2}\|w_i\|^2 + R_{\text{emp}}(w_i) - \frac{\lambda}{2}\|w_t\|^2 - \hat{b}_{[t]}$
12: until $\epsilon_t \leq \epsilon$
```

3.2 Parallelization

Algorithms 2, 3, and 4 the evaluation of $R_{\text{emp}}(w)$ (and $\partial_w R_{\text{emp}}(w)$) is cleanly separated from the computation of new iterate and the choice of regularizer. If R_{emp} is additively decomposable over the examples (x_i, y_i), that is, can be expressed as a sum of some independent loss terms $l(x_i, y_i, w)$, then we can parallelize these algorithms easily by splitting the data sets and the computation R_{emp} over multiple machines. This parallelization scheme not only reduces the computation time but also allows us to handle data set with size exceeding the memory available on a single machine.

Without loss of generality, we describe a parallelized version of Algorithm 2 here. Assume there are p slave machines and 1 master machine available. At the beginning, we partition a given data set $D = \{(x_i, y_i)\}_{i=1}^m$ into p disjoint sub-datasets $\{D_i\}_{i=1}^p$ and assign one sub-dataset to each slave machine. At iteration t, the master first broadcasts the current iterate w_{t-1} to all p slaves (e.g., using MPI function `MPI::Broadcast` Gropp et al. 1999). The slaves then compute the losses and (sub)gradients on their local sub-datasets in parallel. As soon as the losses and (sub)gradients computation finished, the master combines the results (e.g., using `MPI::AllReduce`). With the combined (sub)gradient and offset, the master computes the new iterate w_t as in Algorithms 2 and 3. This process repeats until convergence is achieved. Detailed pseudocode can be found in Algorithm 5.

4. Related Research

The *kernel trick* is widely used to transform many existing machine learning algorithms into ones operating on a Reproducing Kernel Hilbert Space (RKHS). One lifts w into an RKHS and replaces all inner product computations with a positive definite kernel function $k(x, x') \leftarrow \langle x, x' \rangle$. Examples of algorithms which employ the kernel trick (but essentially still solve (1)) include Support Vector regression (Vapnik et al., 1997), novelty detection (Schölkopf et al., 2001), Huber's robust regression, quantile regression (Takeuchi et al., 2006), ordinal regression (Herbrich et al., 2000), rank-

Algorithm 5 Parallel BMRM

```
1: input: $\epsilon \geq 0$, $w_0$, data set $D$, number of slave machines $p$
2: initialization: $t \leftarrow 0$, assign sub-dataset $D_i$ to slave $i$, $i = 1, \ldots, p$
3: repeat
4:     $t \leftarrow t + 1$
5:     Master: Broadcast $w_{t-1}$ to all slaves
6:     Slaves: Computes $R^i_{\mathrm{emp}}(w_{t-1}) := \sum_{(x,y)\in D_i} l(x, y, w_{t-1})$ and $a^i_t \in \partial_w R^i_{\mathrm{emp}}(w_{t-1})$
7:     Master: Aggregate $a_t := \frac{1}{|D|}\sum_{i=1}^p a^i_t$ and $b_t := \frac{1}{|D|}\sum_{i=1}^p R^i_{\mathrm{emp}}(w_{t-1}) - \langle w_{t-1}, a_t\rangle$
8:     Master: Update model $R^{\mathrm{CP}}_t(w) := \max_{1\leq j\leq t}\{\langle w, a_j\rangle + b_j\}$
9:     Master: $w_t \leftarrow \mathrm{argmin}_w J_t(w) := \lambda\Omega(w) + R^{\mathrm{CP}}_t(w)$
10:    Master: $\epsilon_t \leftarrow \min_{0\leq i\leq t} J(w_i) - J_t(w_t)$
11: until $\epsilon_t \leq \epsilon$
12: return $w_t$
```

ing (Crammer and Singer, 2005), maximization of multivariate performance measures (Joachims, 2005), structured estimation (Taskar et al., 2004; Tsochantaridis et al., 2005), Gaussian Process regression (Williams, 1998), conditional random fields (Lafferty et al., 2001), graphical models (Cowell et al., 1999), exponential families (Barndorff-Nielsen, 1978), and generalized linear models (Fahrmeir and Tutz, 1994).

Traditionally, specialized solvers have been developed for solving the kernel version of (1) in the dual (see, e.g., Chang and Lin, 2001; Joachims, 1999). These algorithms construct the Lagrange dual, and solve for the Lagrange multipliers efficiently. Only recently, research focus has shifted back to solving (1) in the primal (see, e.g., Chapelle, 2007; Joachims, 2006; Sindhwani and Keerthi, 2006). This spurt in research interest is due to three main reasons: First, many interesting problems in diverse areas such as text classification, word-sense disambiguation, and drug design already employ rich high dimensional data which does not necessarily benefit from the kernel trick. All these domains are characterized by large data sets (with m in the order of a million) and very sparse features (e.g., the bag of words representation of a document). Second, efficient factorization methods (e.g., Fine and Scheinberg, 2001) can be used for a low rank representation of the kernel matrix thereby effectively rendering the problem linear. Third, approximation methods such as the *Random Feature Map* proposed by Rahimi and Recht (2008) can efficiently approximate a infinite dimensional nonlinear feature map associated to a kernel by a finite dimensional one. Therefore our focus on the primal optimization problem is not only pertinent but also timely.

The widely used SVM$^{\text{struct}}$ optimizer of Thorsten Joachims[5] is closely related to BMRM. While BMRM can handle many different regularizers and loss functions, SVM$^{\text{struct}}$ is mainly geared towards square norm regularizers and non-differentiable soft-margin type loss functions. On the other hand, SVM$^{\text{struct}}$ can handle kernels while BMRM mainly focuses on the primal problem.

Our convergence analysis is closely related to Shalev-Shwartz and Singer (2006) who prove mistake bounds for online algorithms by lower bounding the progress in the dual. Although not stated explicitly, essentially the same technique of lower bounding the dual improvement was used by Tsochantaridis et al. (2005) to show polynomial time convergence of the SVM$^{\text{struct}}$ algorithm. The main difference however is that Tsochantaridis et al. (2005) only work with a quadratic ob-

5. Software available at `http://svmlight.joachims.org/svm_struct.html`.

jective function while the framework proposed by Shalev-Shwartz and Singer (2006) can handle arbitrary convex functions. In both cases, a weaker analysis led to $O(1/\varepsilon^2)$ rates of convergence for nonsmooth loss functions. On the other hand, our results establish a $O(1/\varepsilon)$ rate for nonsmooth loss functions and $O(\log(1/\varepsilon))$ rates for smooth loss functions under mild technical assumptions.

Another related work is SVM$^{\text{perf}}$ (Joachims, 2006) which solves the SVM with linear kernel in linear time. SVM$^{\text{perf}}$ finds a solution with accuracy ε in $O(md/(\lambda\varepsilon^2))$ time, where the m training patterns $x_i \in \mathbb{R}^d$. This bound was improved by Shalev-Shwartz et al. (2007) to $\tilde{O}(1/\lambda\delta\varepsilon)$ for obtaining an accuracy of ε with confidence $1-\delta$. Their algorithm, Pegasos, essentially performs stochastic (sub)gradient descent but projects the solution back onto the L_2 ball of radius $1/\sqrt{\lambda}$. Note that Pegasos also can be used in an online setting. This, however, only applies whenever the empirical risk decomposes into individual loss terms (e.g., it is not applicable to multivariate performance scores Joachims 2005).

The third related strand of research considers gradient descent in the primal with a line search to choose the optimal step size (see, e.g., Boyd and Vandenberghe, 2004, Section 9.3.1). Under assumptions of smoothness and strong convexity – that is, the objective function can be upper and lower bounded by quadratic functions – it can be shown that gradient descent with line search will converge to an accuracy of ε in $O(\log(1/\varepsilon))$ steps. Our solver achieves the same rate guarantees for smooth functions, under essentially similar technical assumptions.

We would also like to point out connections to subgradient methods (Nedich and Bertsekas, 2000). These algorithms are designed for nonsmooth functions, and essentially choose an arbitrary element of the subgradient set to perform a gradient descent like update. Let $\max_{u \in \partial_w J(w)} \|u\| \leq G$, and $B(w^*, r)$ denote a ball of radius r centered around the minimizer of $J(w)$. By applying the analysis of Nedich and Bertsekas (2000) to the regularized risk minimization problem with $\Omega(w) := \frac{\lambda}{2}\|w\|^2$, Ratliff et al. (2007) show that subgradient descent with a fixed, but sufficiently small, stepsize will converge linearly to $B(w^*, G/\lambda)$.

Finally, several papers (Keerthi and DeCoste, 2005; Chapelle, 2007) advocate the use of Newton-like methods to solve Support Vector Machines in the "primal". However, they need to take precautions when dealing with the fact that the soft-margin type of loss functions such as the hinge loss is only piecewise differentiable. Instead, our method only requires *subdifferentials*, which always exist for convex functions, in order to make progress. The large number of and variety of implemented problems shows the flexibility of our approach.

5. Experiments

In this section, we examine the convergence behavior of BMRM and show that it is versatile enough to solve a variety of machine learning problems. All our experiments were carried out on a cluster of 24 machines each with a 2.4GHz AMD Dual Core processor and 4GB of RAM. Details of the loss functions, data sets, competing solvers and experimental objectives are described in the following subsections.

5.1 Convergence Behavior

We investigated the convergence rate of our method (Algorithm 2) empirically with respect to regularization constant λ, approximation gap ε, and bundle size k. In addition, we investigated the speedup gained by parallelizing the empirical risk computation. Finally, we examined empirically how generalization performance is related to approximation gap. For simplicity, we focused on the training of a linear SVM with binary hinge loss:[6]

$$\min_w J(w) := \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} \max(0, 1 - y_i \langle w, x_i \rangle). \tag{15}$$

The experiments were conducted on 6 data sets commonly used in binary classification studies, namely, adult9, astro-ph, news20-b,[7] rcv1, real-sim, and worm. adult9, news20-b, rcv1, and real-sim are available on the LIBSVM tools website.[8] astro-ph (Joachims, 2006) and worm (Franc and Sonnenburg, 2008) are available upon request from Thorsten Joachims and Soeren Sonnenburg, respectively. Table 1 summarizes the properties of the data sets.

Data Set	#examples m	dimension d	density %
adult9	48,842	123	11.27
astro-ph	94,856	99,757	0.08
news20-b	19,954	1,355,191	0.03
rcv1	677,399	47,236	0.15
real-sim	72,201	20,958	0.25
worm	1,026,036	804	25.00

Table 1: Properties of the binary classification data sets used in our experiments.

5.1.1 REGULARIZATION CONSTANT λ AND APPROXIMATION GAP ε

As suggested by the convergence analysis, the linear SVM with the nonsmooth binary hinge loss should converge in $O(\frac{1}{\lambda\varepsilon})$ iterations, where λ and ε are two parameters which one normally tunes during the model selection phase. Therefore, we investigated the scaling behavior of our method w.r.t. these two parameters. We performed the experiments with unlimited bundle size and with a heuristic that removes subgradients which remained inactive (i.e., Lagrange multiplier = 0) for 10 or more consecutive iterations.[9]

Figure 4 shows the approximation gap ε_t as a function of number of iterations t. As predicted by our convergence analysis, BMRM converges faster for larger values of λ. Furthermore, the empirical convergence curves exhibit a $O(\log(1/\varepsilon))$ rate instead of the (pessimistic) theoretical rate of $O(\frac{1}{\varepsilon})$, especially for large values of λ. Interestingly, BMRM converges faster on high-dimensional text data sets (i.e., astro-ph, news20-b, rcv1, and real-sim) than on lower dimensional data sets (i.e., adult9 and worm).

5.1.2 BUNDLE SIZE

The dual of our method (10) is a concave problem which has dimensionality equal to the number of iterations executed. In the case of linear SVM, (10) is a QP. Hence, as described in Section 3.1, we can trade potentially greater bundle improvement for memory efficiency.

6. Similar behavior was observed with other loss functions.
7. The data set is originally named news20; we renamed it to avoid confusion with the multiclass version of the data set.
8. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`.
9. Note that this heuristic does not have any implication in the convergence analysis.

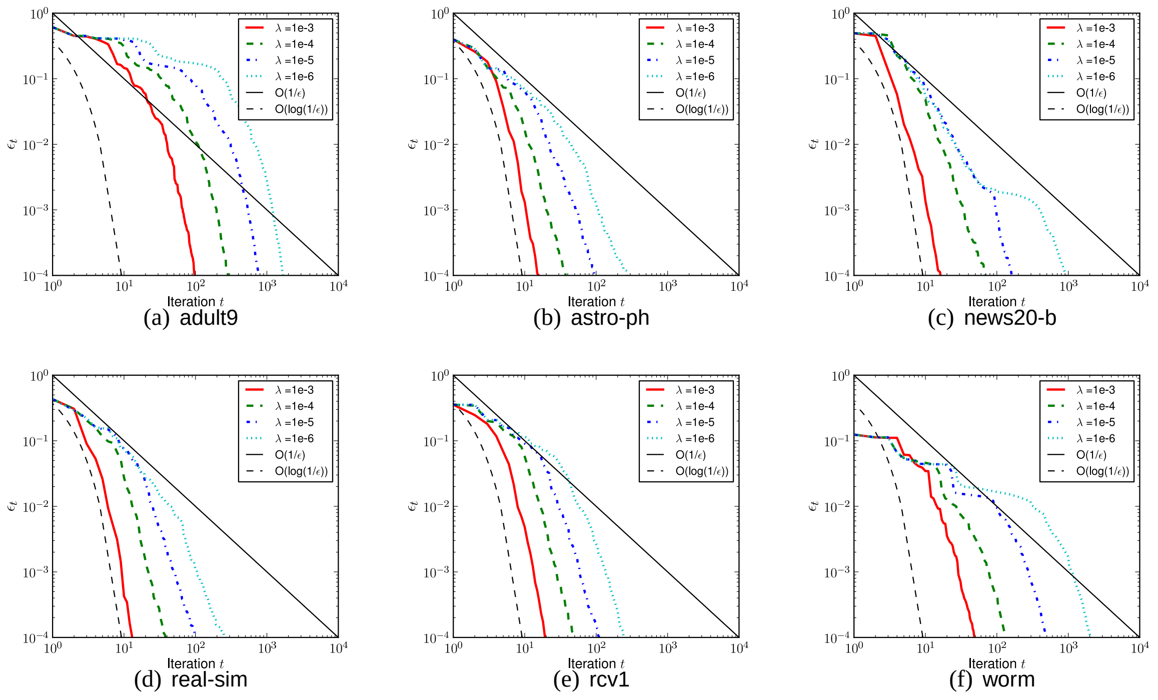


Figure 4: Approximation gap ε_t as a function of number of iterations t; for different regularization constants λ (and unlimited bundle size).

Figure 5 shows the approximation gap ε_t during the training of linear SVM as a function of the number of iterations t, for different bundle sizes $k \in \{2, 10, 50, \infty\}$. In the case of $k = \infty$, we employed the same heuristics which remove inactive linearizations as those mentioned in Section 5.1.1. As expected, the larger k is, the faster the algorithm converges. Although the case $k = 2$ is the slowest, its convergence rate is still faster than the theoretical bound $\frac{1}{\lambda\varepsilon}$.

5.1.3 PARALLELIZATION

When the empirical risk R_{emp} is additively decomposable, the loss and subgradient computation can be executed concurrently on multiple processors for different subsets of data points.[10]

We performed experiments for linear SVMs training with parallelized risk computation on the worm data set. Figure 6(a) shows the wallclock time for the overall training phase (e.g., data loading, risk computation, and solving the QP) and CPU time for just the risk computation as a function of number of processors p. Note that the gap between the two curves essentially tells the runtime upper bound of the sequential part of the algorithm. As expected, both overall and risk computation time decrease as the number of processors p increases. However, in Figure 6(b), we see two different speedups.[11] The speedup for the risk computation is roughly linear as there is no sequential part in

10. This requires only slight modification to the data loading process and the addition of some parallelization related code before and after the code segment for empirical risk computation.

11. Speedup $S_p = \frac{T_1}{T_p}$ where p is the number of processors and T_q is the runtime of the parallelized algorithm on q processors.

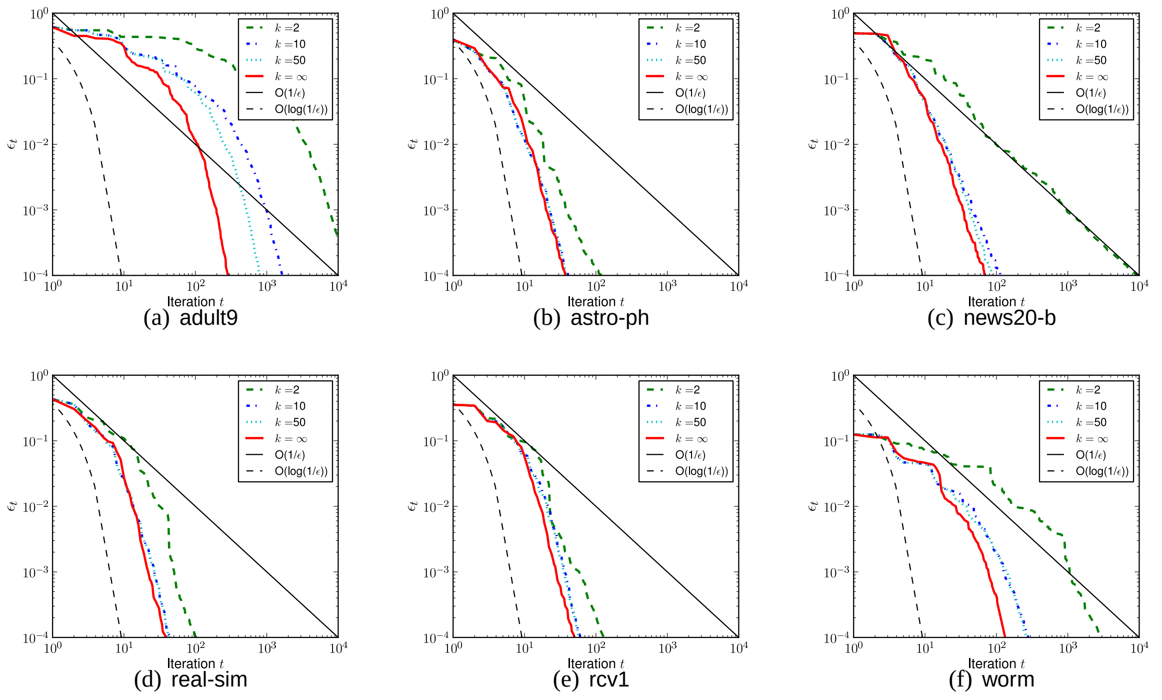


Figure 5: Approximation gap ϵ_t as a function of number of iterations t; for different bundle sizes k (and fixed regularization constant $\lambda = 10^{-4}$).

it; the speedup of overall computation is approaching a limit[12] as well-explained by Amdahl's law (Amdahl, 1967).

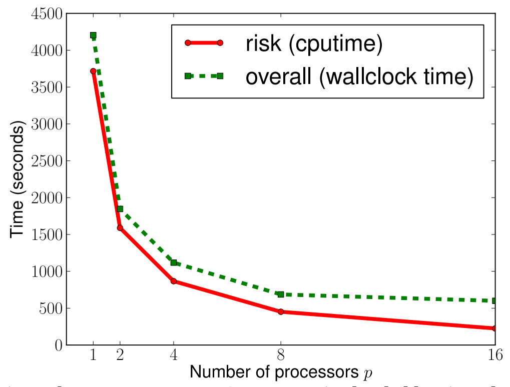
5.1.4 GENERALIZATION VERSUS APPROXIMATION GAP

Since the problems we are considering are convex, all properly convergent optimizers will converge to the same solution. Therefore, comparing generalization performance of the final solution is meaningless. But, in real life one is often interested in the speed with which the algorithm achieves good generalization performance. In this section we study this question. We focus on the generalization (in terms of accuracy) as a function of approximation gap during training. For this experiment, we randomly split each of the data sets into training (60%), validation (20%) and testing (20%) sets.

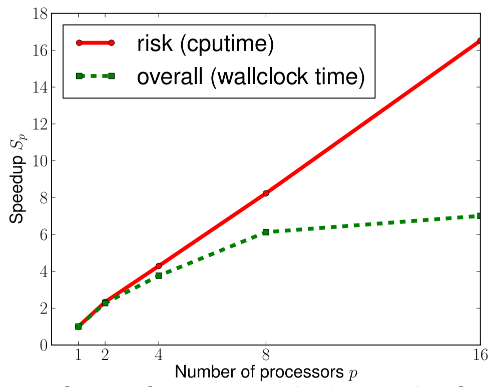
We first obtained the best $\lambda \in \{2^{-20}, \ldots, 2^0\}$ for each of the data sets using their corresponding validation sets. With these best λ's, we (re)trained linear SVMs and recorded the testing accuracy as well as the approximation gap at every iteration, with termination criterion $\epsilon = 10^{-4}$. Figure 7 shows the difference between the testing accuracy evaluated at every iteration and that after training, as a function of approximation gap at each iteration.

From the figure, we see that the testing accuracies for adult9 and worm data sets are less stable in general and the approximation gap must be reduced to at least 10^{-3} to reach the 0.5% regime

12. The limit of speedup is the inverse of the sequential fraction of the algorithm such as the QP.



(a) Risk computation in CPU time (red solid line) and overall computation (i.e., data loading + risk computation + solving the QP) in wallclock time (green dashed line) as a function of number of processors.



(b) Speedup in risk computation (in CPU time) and overall computation (in wallclock time) as a function of number of processors.

Figure 6: CPU and wallclock time for training linear SVM using parallel BMRM on worm data set with varying number of processors $p \in \{1,2,4,8,16\}$. In these experiments, regularization constant $\lambda = 10^{-6}$, and termination criterion $\varepsilon = 10^{-4}$.

of the final testing accuracies; the testing accuracies for the rest of the data sets arrived at the same regime with approximation gap of 10^{-2} or lower.

In general, the generalization improved as the approximation gap decreased. The improvement in generalization became rather insignificant (say, the maximum of changes in testing accuracies is less than 0.1%) when the approximation gap was further reduced to below some effective threshold ε_{eff}; that said, it is not necessary to continue the optimization when $\varepsilon_t \leq \varepsilon_{\text{eff}}$.[13] Since ε_{eff} (or its scale) is not known *a priori* and the asymptotic analysis in Shalev-Schwartz and Srebro (2008) does not reveal the actual scale of ε_{eff} directly applicable in our case, we carried out another set of experiments to investigate if ε_{eff} could be estimated with as little effort as possible: For each data set, we randomly subsampled $10\%,\ldots,50\%$ of the training set as sub-datasets and performed the same experiment on all sub-datasets. We then determined the largest ε_{eff} such that the maximum changes in testing accuracies is less than 0.1%.

Table 5.1.4 shows the (base 10 logarithm of) ε_{eff} for all sub-datasets as well as the full data sets. It seems that the ε_{eff} estimated on a smaller sub-dataset is at most 1 order of magnitude larger than the actual ε_{eff} required on full data set. In addition, we show in the table that the necessary threshold $\varepsilon_{10\%}$ required by the sub-datasets and the full data sets to attain the final testing accuracies attained by the 10% sub-datasets. The observations obey the analysis in Shalev-Schwartz and Srebro (2008) that for a fixed testing accuracy, approximation gap (i.e., optimization error) can be relaxed when more data is given.

13. Heuristically, we could terminate the training phase following the *early stopping* strategy by monitoring the changes in accuracies on validation set evaluated in some most recent iterations.

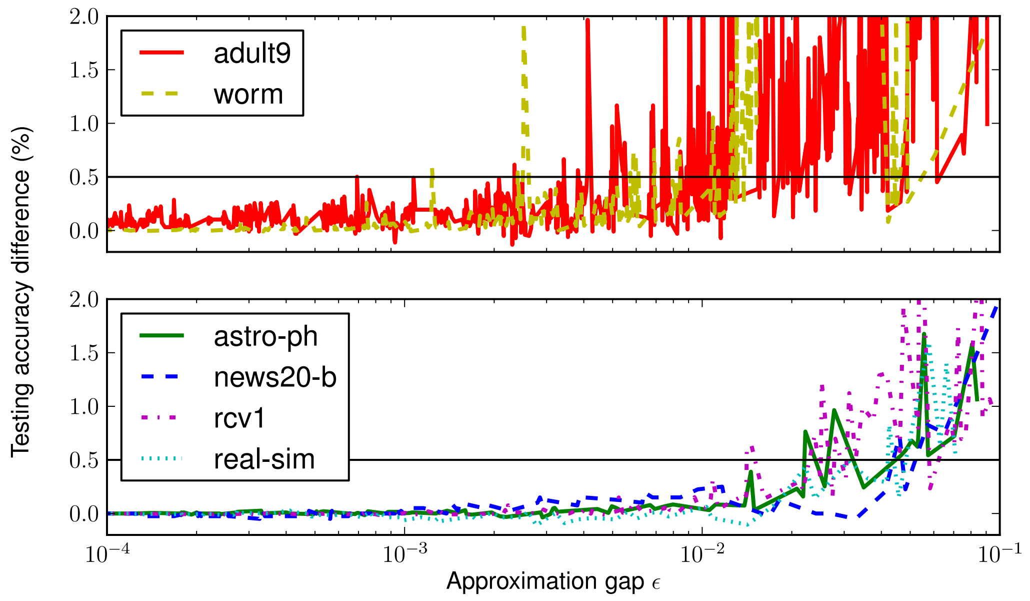


Figure 7: Difference between testing accuracies of intermediate and final models.

5.2 Comparison with Existing Bundle Methods

In this section we compare BMRM with a BT implementation obtained from Schramm and Zowe (1992).[14] We also compare the performance of BMRM (Algorithm 2) and LSBMRM (Algorithm 3). The multiclass line search used in LSBMRM can be found in Yu et al. (2008).

For binary classification, we solve the linear SVM (15) on the data sets: adult9, astro-ph, news20-b, rcv1, real-sim, and worm as mentioned in Section 5.1. For multiclass classification, we solve (Crammer and Singer, 2003):

$$\min_w J(w) := \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} \max_{y_i' \in [c]} \left\langle w, e_{y_i'} \otimes x_i - e_{y_i} \otimes x_i \right\rangle + \mathbf{I}(y_i \neq y_i'), \tag{16}$$

where c is the number of classes in the problem, e_i is the i-th standard basis for $\mathbb{R}^c$, $\otimes$ denotes Kronecker product; and $\mathbf{I}(\cdot)$ is an indicator function that has value 1 if its argument is evaluated true, and 0 otherwise. The data sets used in multiclass classification experiments were inex, letter, mnist, news20-m,[15] protein, and usps. inex is available for download on the website of Antoine Bordes[16] and the rest can be found on the LIBSVM tools website.[17] Table 3 summarizes the properties of these data sets.

In each of the experiments, we first obtain the optimal weight vector $\bar{w}$ by running BMRM until the termination criteria $J(w_t) - J_t(w_t) \leq 0.01 J(w_t)$ is satisfied. Then we run BT, LSBMRM, and

14. The original FORTRAN implementation was automatically converted into C for use in our library.
15. The data set is originally named news20; we renamed it to avoid confusion with the binary version of the data set.
16. Software available at `http://webia.lip6.fr/~bordes/datasets/multiclass/inex.tar.gz`.
17. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html`.

		10%	20%	30%	40%	50%	100%
adult9	Acc. (%)	84.3	84.7	84.9	85.1	85.1	85.2
	$\log_{10}\varepsilon_{\text{eff}}$	-3.90	-3.72	-3.77	-3.88	-3.64	-4.00
	$\log_{10}\varepsilon_{10\%}$	-4.01	-1.18	-1.07	-1.16	-1.27	-1.04
astro-ph	Acc. (%)	96.1	96.6	96.4	96.6	96.8	97.4
	$\log_{10}\varepsilon_{\text{eff}}$	-1.48	-1.70	-1.57	-1.49	-1.68	-1.84
	$\log_{10}\varepsilon_{10\%}$	-4.00	-1.15	-1.06	-0.98	-1.02	-0.87
news20-b	Acc. (%)	89.9	92.9	94.3	94.5	95.4	96.6
	$\log_{10}\varepsilon_{\text{eff}}$	-2.00	-2.48	-3.87	-1.65	-3.71	-2.84
	$\log_{10}\varepsilon_{10\%}$	-4.02	-0.92	-0.70	-0.80	-0.80	-0.67
rcv1	Acc. (%)	96.9	97.2	97.4	97.2	97.5	97.6
	$\log_{10}\varepsilon_{\text{eff}}$	-2.02	-2.40	-1.99	-2.16	-2.34	-2.28
	$\log_{10}\varepsilon_{10\%}$	-4.07	-1.19	-1.30	-1.29	-1.13	-1.11
real-sim	Acc. (%)	95.0	95.9	96.3	96.6	96.6	97.2
	$\log_{10}\varepsilon_{\text{eff}}$	-1.74	-1.84	-1.71	-1.99	-1.74	-1.75
	$\log_{10}\varepsilon_{10\%}$	-4.02	-1.04	-0.88	-0.87	-0.85	-0.82
worm	Acc. (%)	98.2	98.2	98.2	98.3	98.3	98.4
	$\log_{10}\varepsilon_{\text{eff}}$	-2.43	-2.47	-2.48	-3.62	-2.81	-3.55
	$\log_{10}\varepsilon_{10\%}$	-4.00	-1.38	-1.28	-1.37	-1.28	-1.31

Table 2: The first sub-row in each data set row indicates the testing accuracies of models trained on the corresponding proportions of the training set. The second sub-row indicates the (base 10 logarithm of) effective threshold such that the maximum difference in testing accuracies of models with approximation gap smaller than that is less than 0.1%. The third sub-row indicates the (base 10 logarithm of) threshold necessary for models to attain the testing accuracy attained by the model trained on the 10% sub-dataset with default $\varepsilon = 10^{-4}$.

Data Set	#examples m	#classes c	dimension d	density %
inex	12,107	18	167,295	0.48
letter	20,000	26	16	100.00
mnist	70,000	10	780	19.24
news20-m	19,928	20	62,061	0.13
protein	21,516	3	357	28.31
usps	9,298	10	256	96.70

Table 3: Properties of the multiclass classification data sets used in the experiments.

BMRM until the following termination criteria is satisfied:

$$J(w_t) - J(\bar{w}) \leq 0.01 J(\bar{w}). \tag{17}$$

Figure 8 shows the number of iterations t required by the three methods on each data set to satisfy (17) as a function of regularization constant $\lambda \in \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. As expected, LSBMRM, which uses an exact line search, outperformed both BMRM and BT on all data sets. BMRM performed better than BT on all high dimensional data sets except news20-m but worse on the rest.

Although BT tunes the stabilization trade-off parameter κ_t automatically, it still does not guarantee superiority over BMRM which is considerably simpler. Nevertheless, external stabilization (in BT) clearly helps speed up the convergence in certain cases.

5.3 Versatility

In the following subsections, we will illustrate some of the applications of BMRM to various machine learning problems with smooth and non-differentiable loss functions, and with different regularizers. Our aim is to show that BMRM is versatile enough to be used in a variety of seemingly different problems. Readers not interested in this aspect of BMRM can safely skip this subsection.

5.3.1 BINARY CLASSIFICATION

In this section, we evaluate the performance of our method BMRM in the training of binary classifier using linear SVMs (15) and logistic loss:

$$\min_w J(w) := \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} \log(1+\exp(-y_i\langle w, x_i\rangle)),$$

on the binary classification data sets mentioned in Section 5.1 with split similar to that in Section 5.1.4. Since we will compare BMRM with other solvers which use different termination criteria, we consider the CPU time used in reducing the relative difference between the current smallest objective function value and the optimum:

$$\frac{\min_{i\le t} J(w_i) - J(w^*)}{J(w^*)},$$

where w_i is the weight vector at time/iteration i, and w^* is the minimizer obtained by running BMRM until the approximation gap $\varepsilon_t < 10^{-4}$. The best $\lambda \in \left\{2^{-20}, \ldots, 2^0\right\}$ for each of the data sets was determined by evaluating the performance on the corresponding validation set.[18]

In the case of linear SVMs, we compared BMRM to three publicly available state of the art *batch learning* solvers:

1. OCAS (Franc and Sonnenburg, 2008). Since this method is equivalent to LSBMRM with binary hinge loss, we refer to this software by LSBMRM for naming consistency.
2. LIBLINEAR (Fan et al., 2008) version 1.33 with option "-s 3".
3. SVM$^{\text{perf}}$ (Joachims, 2006) version 2.5 with option "-w 3" and with double precision floating point numbers.

LIBLINEAR solves the dual problem of linear SVM using a coordinate descent method (Hsieh et al., 2008). SVM$^{\text{perf}}$ was chosen for comparison as it is algorithmically identical to BMRM in this case. Both LIBLINEAR and SVM$^{\text{perf}}$ provide a "shrinking" technique to speed up the algorithms by ignoring some data points which are not likely to affect the objective. Since BMRM does not provide such shrinking technique, we excluded this option in both LIBLINEAR and SVM$^{\text{perf}}$ for a fair comparison.

Figure 9 shows the relative difference in objective value as a function of training time (CPU seconds) for three methods on various data sets. BMRM is faster than SVM$^{\text{perf}}$ on all data sets

18. The corresponding penalty parameter C for LIBLINEAR and OCAS is $1/(m\lambda)$, and for SVM$^{\text{perf}}$ is $1/(100\lambda)$.

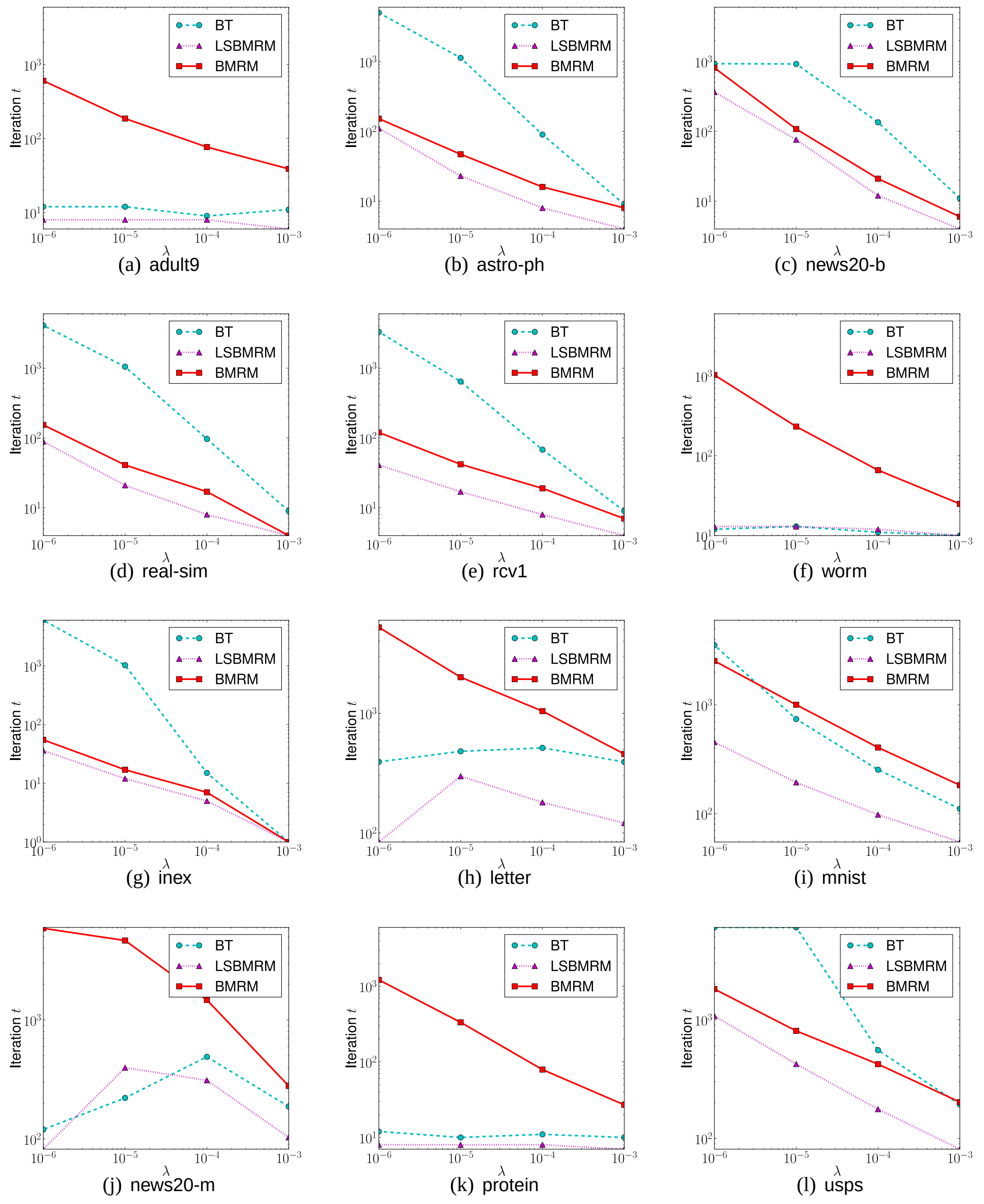


Figure 8: Smallest number of iterations required to satisfy the termination criterion (17) for each data set and various regularization constants. (BT did not satisfy (17) in the inex and usps experiments for $\lambda = 10^{-6}$ after 6000 iterations.)

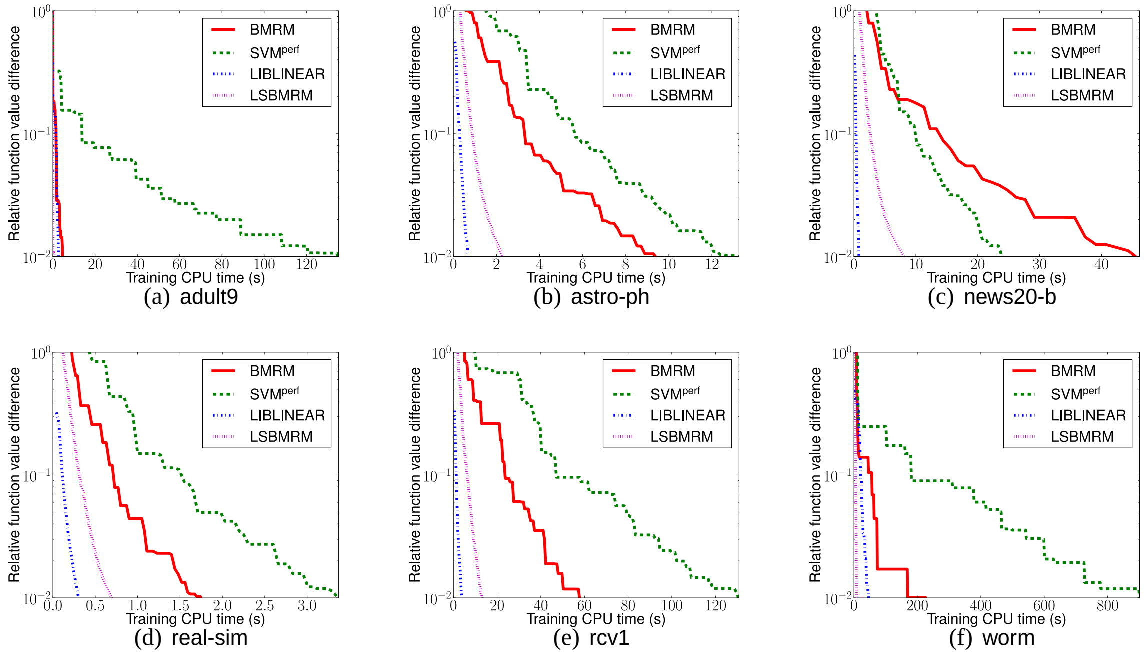


Figure 9: Linear SVMs. Relative primal objective value difference during training.

except news20-b. The performance difference observed here is largely due to the differences in the implementations (e.g., feature vector representation, QP solver, etc.). Nevertheless, both BMRM and SVM^{perf} are significantly outperformed by LSBMRM and LIBLINEAR on all data sets, and LIBLINEAR is almost always faster than LSBMRM. It is clear from the figure that LSBMRM and LIBLINEAR enjoy progression with "strictly" decreasing objective values; whereas the progress of both BMRM and SVM^{perf} are hindered by the "stalling" steps (i.e., the flat line segments in the plots). The fact that LSBMRM is different from BMRM and SVM^{perf} by one additional line search step implies that the "stalling" steps is the time that BMRM and SVM^{perf} improve the approximation at the regions which do not help reducing the primal objective function value.

In the case of logistic regression, we compare BMRM to the state of the art trust region Newton method for logistic regression (Lin et al., 2008) which is also available in the LIBLINEAR package (option "-s 0"). From Figure 10, we see that LIBLINEAR outperforms BMRM on all data sets and that BMRM suffers from the same "stalling" phenomenon as observed in the linear SVMs case.

5.3.2 LEARNING THE COST MATRIX FOR GRAPH MATCHING

In computer vision, there are problems which require matching the objects of interest in a pair of images. These problems are often modeled as attributed graph matching problems where the (extracted) landmark points x_i in the first image x must be matched to the corresponding points $x'_{i'}$ in the second image x'. Note that we represent the *point* x_i or $x'_{i'}$ as d-dimensional feature vectors. The attributed graph matching problem is then cast as a *Linear Assignment Problem* (LAP) which

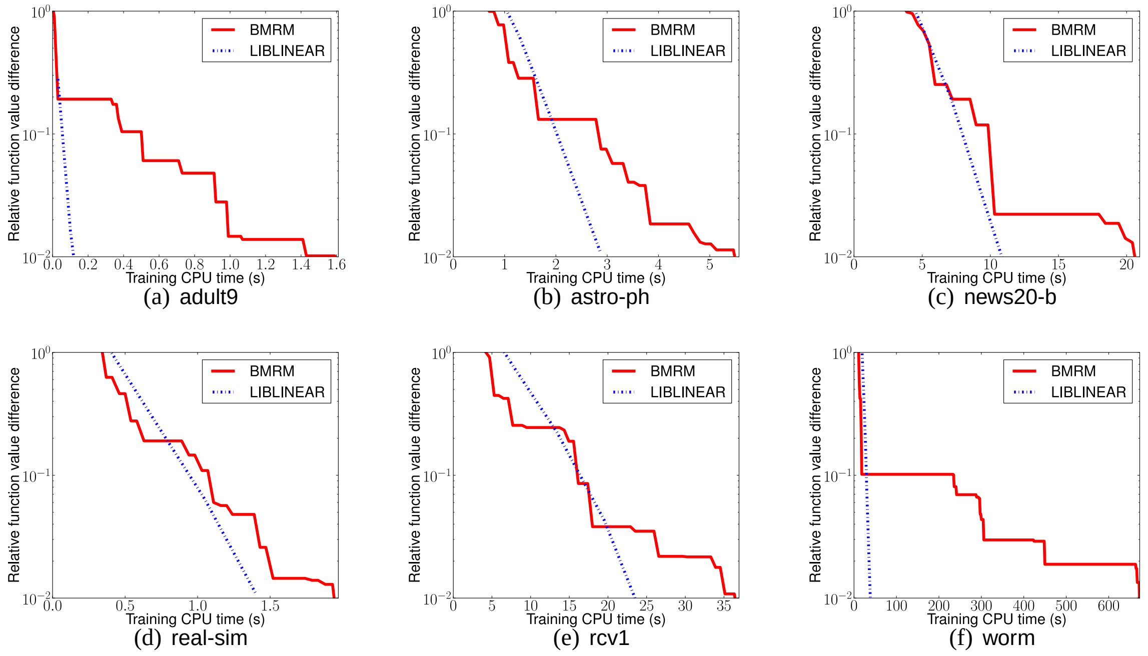


Figure 10: Logistic regression. Relative primal objective value difference during training.

can be solved in worst case $O(n^3)$ time where n is the number of landmark points (Kuhn, 1955).[19] Formally, the LAP reads

$$\max_{y \in \mathcal{Y}} \sum_{i=1}^{n} \sum_{i'=1}^{n} y_{ii'} C_{ii'},$$

where $\mathcal{Y}$ is the set of all $n \times n$ permutation matrices, and $C_{ii'}$ is the cost of matching point x_i to point $x'_{i'}$. In the standard setting of graph matching, one way to determine the cost matrix C is as

$$C_{ii'} := -\sum_{k=1}^{d} \left| x_i^{(k)} - x_{i'}^{\prime(k)} \right|^2.$$

Instead of finding more features to describe the points x_i and $x'_{i'}$ that might improve the matching results, Caetano et al. (2007) propose to learn a weighting to a given set of features that actually improved the matching results in many cases (Caetano et al., 2008).

In Caetano et al. (2007, 2008) the problem of learning the cost matrix for graph matching is formulated as a L_2 regularized risk minimization with loss function

$$l(x, x', y, w) = \max_{\bar{y} \in \mathcal{Y}} \left\langle w, \phi(x, x', \bar{y}) - \phi(x, x', y) \right\rangle + \Delta(\bar{y}, y), \tag{18}$$

where the feature map ϕ is defined as

$$\phi(x, x', y) = -\sum_{i=1}^{n} \sum_{i'=1}^{n} y_{ii'} \left(|x_i^{(1)} - x_{i'}^{\prime(1)}|^2, \ldots, |x_i^{(d)} - x_{i'}^{\prime(d)}|^2 \right), \tag{19}$$

19. To achieve better matching results, one could further enforce edge-to-edge matching where edge refers to pair of landmark points. This additional matching requirement renders the problem as a *Quadratic Assignment Problem*.

and the label loss Δ is defined as the normalized Hamming loss

$$\Delta(\bar{y},y) = 1 - \frac{1}{n}\sum_{i=1}^{n}\sum_{i'=1}^{n}\bar{y}_{ii'}y_{ii'}. \tag{20}$$

By (19) and (20), the argument of (18) becomes

$$\left\langle w, \phi(x,x',\bar{y}) - \phi(x,x',y)\right\rangle + \Delta(\bar{y},y) = \sum_{i=1}^{n}\sum_{i'=1}^{n}\bar{y}_{ii'}\tilde{C}_{ii'} + \text{constant},$$

where $\tilde{C}_{ii'} = -\sum_{k=1}^{d} w_k |x_i^{(k)} - x_{i'}^{\prime(k)}|^2 - y_{ii'}/n$. Therefore, (18) is exactly a LAP. We refer interested readers to Caetano et al. (2007, 2008) for more detailed exposition especially on the use of edge matching (in addition to point matching) which leads to much better performance.

We reproduced the experiment in Caetano et al. (2008) that used BMRM with L_2 regularization on the CMU house data set.[20] For this data set, there are 30 hand-labeled landmark points in each image and the features for those points are the 60-dimensional Shape Context features (Belongie et al., 2001; Caetano et al., 2008). The experiments evaluated the performance of the method for training/validation/testing pairs fixed at baselines (separation of frames) $0, 10, \ldots, 90$. Additionally, we ran the same set of experiments with L_1 regularization, that is, $\Omega(w) = \|w\|_1$.[21] The matching performance of the cost matrices augmented with learned weight vectors w's are compared with the original non-learning cost matrix, that is, with uniform weight vector $w = (1, \ldots, 1)$.

Figure 11 shows the results of the experiments. On the left, we see that the matching performance with learned cost matrices are getting more superior to that of non-learning as the baseline increases. The performance of L_1 and L_2 regularized learning are quite similar on average. On the right are the best learned weights for the features using L_1 regularization (top) and L_2 regularization (bottom) for baseline 50. The weights due to L_1 regularization is considerably sparser (i.e., 42 non-zeros) than that due to L_2 regularization (i.e., 52 non-zeros).

5.3.3 HUMAN ACTION SEGMENTATION AND RECOGNITION

In this section, we consider the problem of joint segmentation and recognition of human action from a video sequence using the discriminative Semi-Markov Models (SMM) proposed by Shi et al. (2008). Denote by $x = \{x_i\}_{i=1}^{n} \in \mathcal{X}$ a sequence of n video frames, and by $y = \{(s_i, c_i)\}_{i=1}^{\bar{n}} \in \mathcal{Y}$ the corresponding segment labeling where s_i is the starting location of the i-th segment which ends at $s_{i+1} - 1$, c_i is the frame label for all frames in the segment, and $\bar{n} \leq n$ the number of segments. For ease of presentation, we append a dummy video frame x_{n+1} to x and a dummy segment label $(s_{\bar{n}+1}, c_{\bar{n}+1})$ to y to mark x_{n+1} as the last segment.

In SMM, there exists a *segment* variable for each possible segment (i.e., multiple frames) of x that model the frame label and the boundaries (or length) of a segment jointly; these *segment* variables form a Markov Chain. On the contrary, the Hidden Markov Model (HMM) for the same video x has one *frame label* variable y_i for each video frame x_i. The fact that SMM models multiple frames as one variable allows one to exploit the *structure* and *information* in the problem more efficiently than in HMM. The *structure* exploitation is due to the fact that one human action usually

20. This data set consist of a sequence of 111 images of a toy house. Available at `http://vasc.ri.cmu.edu/idb/html/motion/house/index.html`.
21. Further description on L_1 regularized BMRM can be found in Appendix C.

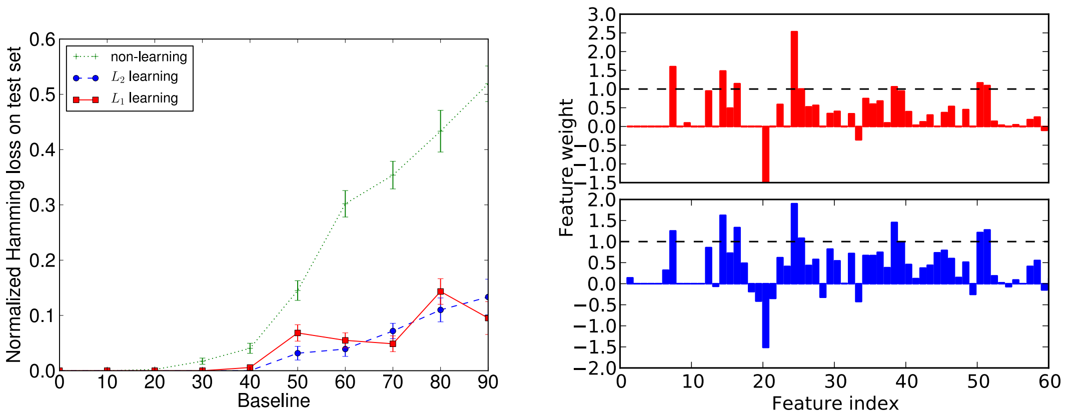


Figure 11: **Left:** Performance on the house data set as the baseline varies. For each baseline, the minimizer of validation loss is evaluated on all testing examples. The corresponding mean normalized Hamming losses (as points) and its standard errors (as error bars) are reported. **Right:** Feature weights for best models trained with L_1 regularization (top) and L_2 regularization (bottom) for baseline 50. Dashed lines indicate the feature weight value 1.

spans several consecutive frames, and the *information* exploitation is due to the possibility to extract features which only become apparent within a segment of several frames.

The discriminative SMM in Shi et al. (2008) is formulated as a regularized risk minimization problem where the loss function is

$$l(x,y,w) = \max_{\bar{y}\in\mathcal{Y}} \langle w, \phi(x,\bar{y}) - \phi(x,y)\rangle + \Delta(\bar{y},y). \tag{21}$$

The feature map ϕ is defined as

$$\phi(x,y) = \left(\sum_{i=1}^{\bar{n}} \phi_1(x,s_i,c_i), \sum_{i=1}^{\bar{n}} \phi_2(x,s_i,s_{i+1},c_i), \sum_{i=1}^{\bar{n}} \phi_3(x,s_i,s_{i+1},c_i,c_{i+1})\right),$$

where ϕ_1, ϕ_2, and ϕ_3 are some feature functions for the segment boundaries, segments, and adjacent segments, respectively. Let y_i be the frame label for x_i according to segment labeling y, the label loss function Δ is defined as

$$\Delta(\bar{y},y) = \sum_{i=1}^{n} \mathbf{I}(\bar{y}_i \neq y_i), \tag{22}$$

where $\mathbf{I}(\cdot)$ is an indicator function as defined in (16). We refer interested readers to Shi et al. (2008) for more details on the features and the dynamic programming to compute (21) and its subgradient.

We followed the experimental setup of Shi et al. (2008) by running BMRM for this problem with L_2 (i.e., $\Omega(w) = \frac{1}{2}\|w\|^2$) and L_1 (i.e., $\Omega(w) = \|w\|_1$) regularization, on the Walk-Bend-Draw (WBD) data sets (Shi et al., 2008) which consists of 18 video sequences with 3 human action classes (i.e., *walking*, *bending*, *drawing*). For this data set, the dimensionality of the image of the feature map ϕ is $d = 9917$.

Table 4 shows the 6 fold cross validation results for our methods (L_1 and L_2 SMM),[22] SVMs and SVM-HMM (Tsochantaridis et al., 2005). The latter two are adopted from Shi et al. (2008). SMM outperforms SVM-HMM and SVM as reported in Shi et al. (2008). Amongst L_1 and L_2 SMMs, the latter performs the best and converged to optimal. Although L_1 SMM failed to satisfy the termination criterion, the performance is comparable to that of L_2 SMM even with a 40 times sparser weight vector (see Figure 12 for the feature weights distributions of L_1 and L_2 SMMs).

Methods	CV mean (std. err.)	#iter.	CPU seconds	nnz(w)
L_2 SMM	0.954 (0.006)	231	1129	3690
L_1 SMM	0.930 (0.010)	500	2659	84
SVM-HMM	0.870 (0.020)	–	–	–
SVM	0.840 (0.030)	–	–	–

Table 4: Experimental results on WBD data set. The second column indicates the mean and standard error of the test accuracy (22). The third and fourth columns indicate the number of iterations and CPU seconds for the training of the final model with the best parameter, and the last column indicates the number of nonzero in the final weight vector w.

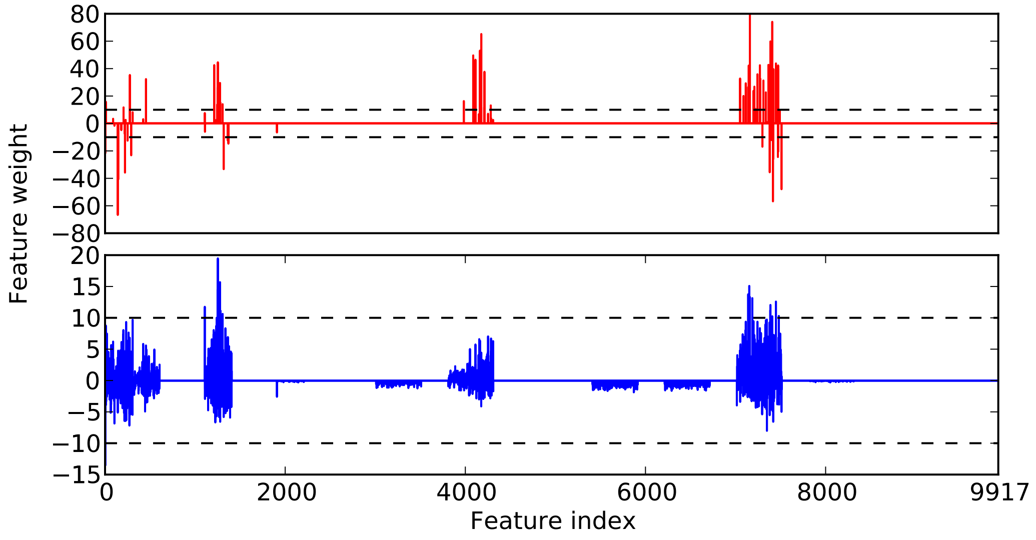


Figure 12: Feature weights for best models trained with L_1 regularization (top) and L_2 regularization (bottom). Dashed lines indicate the feature weight range [$\pm$10].

22. We set termination criterion $\epsilon = 10^{-4}$ and limited the maximum number of iteration to 500.

6. Discussion and Conclusion

The experiments presented in the paper indicate that BMRM is suitable for a wide variety of machine learning problems. In fact, the *modularity* of BMRM not only brings the benefits of parallel and distributed computation but also makes BMRM a natural test bed for trying out new models/ideas on any particular problem with less effort, that is, the user is only required to implement the loss functions and/or regularizers corresponding to different models/ideas.

Nevertheless, we saw in the experiments that BMRM does not guarantee strict improvement in the primal when the dual is solved instead. This phenomenon could significantly hinder the performance of BMRM as seen in some of the experiments. Since efficient line search procedure may not exist for general structured prediction tasks, the *trust region* philosophy used in BT could be a potential strategy to alleviate this problem; we leave this to the future work. We also note that for computationally expensive nonsmooth loss functions, one way to make fuller use of each loss function evaluation is by updating the model R_t^{CP} with two or more linearizations at a non-diffferentiable location (Frangioni, 1997).

In conclusion, we have presented a variant of standard bundle methods, that is, BMRM, which is algorithmically simpler and, in some senses, more straightforward for regularized risk minimization problems than the standard bundle methods. We also showed a $O(1/\varepsilon)$ rate of convergence for nonsmooth objective functions and $O(\log(1/\varepsilon))$ rates for smooth objective functions.

Acknowledgments

NICTA is funded by the Australian Government's Backing Australia's Ability and the Center of Excellence programs. This work is also supported by the IST Program of the European Community, under the FP7 Network of Excellence, ICT-216886-NOE. We thank Jin Yu for many useful discussions and help with implementation issues. Julian McAuley and Qinfeng Shi provided their source code, facilitating some of the experimental comparisons reported in Sections 5.3.2 and 5.3.3. We thank Manfred Warmuth for pointing out Lemma 13. We also thank Sam Roweis, Yasemin Altun, and Nic Schraudolph for insightful discussions. A short, early version of this work appeared in Teo et al. (2007), while the convergence proofs first appeared in Smola et al. (2007).

Appendix A. Loss Functions

A multitude of loss functions are commonly used to derive seemingly different algorithms. This often blurs the similarities as well as subtle differences between them, often for historic reasons: Each new loss is typically accompanied by at least one publication dedicated to it. In many cases, the loss is not spelled out explicitly either but instead, it is only given by means of a constrained optimization problem. A case in point are the papers introducing (binary) hinge loss (Bennett and Mangasarian, 1992; Cortes and Vapnik, 1995) and structured loss (Taskar et al., 2004; Tsochantaridis et al., 2005). Likewise, a geometric description obscures the underlying loss function, as in novelty detection (Schölkopf et al., 2001).

In this section we give an expository yet unifying presentation of many of those loss functions. Many of them are well known, while others, such as multivariate ranking, hazard regression, or Poisson regression are not commonly used in machine learning. Tables 5 and 6 contain a choice subset of simple scalar and vectorial losses. Our aim is to put the multitude of loss functions in

Table 5: Scalar loss functions and their derivatives, depending on $f := \langle w, x \rangle$, and y.

	Loss $\bar{l}(f,y)$	Derivative $\bar{l}'(f,y)$
Hinge (Bennett and Mangasarian, 1992)	$\max(0, 1-yf)$	0 if $yf \geq 1$ and $-y$ otherwise
Squared Hinge (Keerthi and DeCoste, 2005)	$\frac{1}{2}\max(0, 1-yf)^2$	0 if $yf \geq 1$ and $f-y$ otherwise
Exponential (Cowell et al., 1999)	$\exp(-yf)$	$-y\exp(-yf)$
Logistic (Collins et al., 2000)	$\log(1+\exp(-yf))$	$-y/(1+\exp(-yf))$
Novelty (Schölkopf et al., 2001)	$\max(0, \rho - f)$	0 if $f \geq \rho$ and -1 otherwise
Least mean squares (Williams, 1998)	$\frac{1}{2}(f-y)^2$	$f-y$
Least absolute deviation	$\lvert f-y \rvert$	$\operatorname{sgn}(f-y)$
Quantile regression (Koenker, 2005)	$\max(\tau(f-y), (1-\tau)(y-f))$	τ if $f > y$ and $\tau - 1$ otherwise
ε-insensitive (Vapnik et al., 1997)	$\max(0, \lvert f-y \rvert - \varepsilon)$	0 if $\lvert f-y \rvert \leq \varepsilon$, else $\operatorname{sgn}(f-y)$
Huber's robust loss (Müller et al., 1997)	$\frac{1}{2}(f-y)^2$ if $\lvert f-y \rvert \leq 1$, else $\lvert f-y \rvert - \frac{1}{2}$	$f-y$ if $\lvert f-y \rvert \leq 1$, else $\operatorname{sgn}(f-y)$
Poisson regression (Cressie, 1993)	$\exp(f) - yf$	$\exp(f) - y$

Table 6: Vectorial loss functions and their derivatives, depending on the vector $f := Wx$ and on y.

	Loss	Derivative
Soft-Margin Multiclass (Taskar et al., 2004) (Crammer and Singer, 2003)	$\max_{y'}(f_{y'} - f_y + \Delta(y,y'))$	$e_{y^*} - e_y$ where y^* is the argmax of the loss
Scaled Soft-Margin Multiclass (Tsochantaridis et al., 2005)	$\max_{y'} \Gamma(y,y')(f_{y'} - f_y + \Delta(y,y'))$	$\Gamma(y,y')(e_{y^*} - e_y)$ where y^* is the argmax of the loss
Softmax Multiclass (Cowell et al., 1999)	$\log \sum_{y'} \exp(f_{y'}) - f_y$	$\left[\sum_{y'} e_{y'} \exp(f'_y)\right] / \sum_{y'} \exp(f'_y) - e_y$
Multivariate Regression	$\frac{1}{2}(f-y)^\top M(f-y)$ where $M \succeq 0$	$M(f-y)$

an unified framework, and to show how these losses and their (sub)gradients can be computed efficiently for use in our solver framework.

Note that not all losses, while convex, are continuously differentiable. In this situation we give *a* subgradient. While this may not be optimal, the convergence rates of our algorithm do not depend on which element of the subdifferential we provide: in all cases the first order Taylor approximation is a lower bound which is tight at the point of expansion.

In this setion, with little abuse of notation, v_i is understood as the i-th component of vector v when v is clearly not an element of a sequence or a set.

A.1 Scalar Loss Functions

It is well known (Wahba, 1997) that the convex optimization problem

$$\min_{\xi} \xi \text{ subject to } y\langle w,x\rangle \geq 1-\xi \text{ and } \xi \geq 0$$

takes on the value $\max(0, 1-y\langle w,x\rangle)$. The latter is a convex function in w and x. Likewise, we may rewrite the ε-insensitive loss, Huber's robust loss, the quantile regression loss, and the novelty detection loss in terms of loss functions rather than a constrained optimization problem. In all cases, $\langle w,x\rangle$ will play a key role insofar as the loss is convex in terms of the *scalar* quantity $\langle w,x\rangle$. A large number of loss functions fall into this category, as described in Table 5. Note that not all functions of this type are continuously differentiable. In this case we adopt the convention that

$$\partial_x \max(f(x),g(x)) = \begin{cases} \partial_x f(x) & \text{if } f(x) \geq g(x) \\ \partial_x g(x) & \text{otherwise }. \end{cases}$$

Since we are only interested in obtaining an arbitrary element of the subdifferential this convention is consistent with our requirements.

Let us discuss the issue of efficient computation. For all scalar losses we may write $l(x,y,w) = \bar{l}(\langle w,x\rangle, y)$, as described in Table 5. In this case a simple application of the chain rule yields that $\partial_w l(x,y,w) = \bar{l}'(\langle w,x\rangle, y) \cdot x$. For instance, for squared loss we have

$$\bar{l}(\langle w,x\rangle, y) = \tfrac{1}{2}(\langle w,x\rangle - y)^2 \text{ and } \bar{l}'(\langle w,x\rangle, y) = \langle w,x\rangle - y.$$

Consequently, the derivative of the empirical risk term is given by

$$\partial_w R_{\text{emp}}(w) = \frac{1}{m}\sum_{i=1}^{m} \bar{l}'(\langle w,x_i\rangle, y_i) \cdot x_i.$$

This means that if we want to compute l and $\partial_w l$ on a large number of observations x_i, represented as matrix X, we can make use of fast linear algebra routines to pre-compute the vectors

$$f = Xw \text{ and } g^\top X \text{ where } g_i = \bar{l}'(f_i, y_i).$$

This is possible for any of the loss functions listed in Table 5, and many other similar losses. The advantage of this unified representation is that implementation of each individual loss can be done in very little time. The computational infrastructure for computing Xw and $g^\top X$ is shared. Evaluating $\bar{l}(f_i,y_i)$ and $\bar{l}'(f_i,y_i)$ for all i can be done in $O(m)$ time and it is not time-critical in comparison to the remaining operations. Algorithm 6 describes the details.

Algorithm 6 ScalarLoss(w, X, y)

```
1: input: Weight vector $w$, feature matrix $X$, and labels $y$
2: Compute $f = Xw$
3: Compute $r = \sum_i \bar{l}(f_i, y_i)$ and $g = \bar{l}'(f, y)$
4: $g \leftarrow g^\top X$
5: return Risk $r$ and gradient $g$
```

An important but often neglected issue is worth mentioning. Computing f requires us to *right* multiply the matrix X with the vector w while computing g requires the *left* multiplication of X with the vector $g^\top$. If X is stored in a row major format then Xw can be computed rather efficiently while $g^\top X$ is expensive. This is particularly true if X cannot fit in main memory. Converse is the case when X is stored in column major format. Similar problems are encountered when X is a sparse matrix and stored in either compressed row format or in compressed column format.

A.2 Structured Loss

In recent years structured estimation has gained substantial popularity in machine learning (Tsochantaridis et al., 2005; Taskar et al., 2004; Bakir et al., 2007). At its core it relies on two types of convex loss functions: logistic loss:

$$l(x, y, w) = \log \sum_{y' \in \mathcal{Y}} \exp\left(\left\langle w, \phi(x, y')\right\rangle\right) - \left\langle w, \phi(x, y)\right\rangle, \tag{23}$$

and soft-margin loss:

$$l(x, y, w) = \max_{y' \in \mathcal{Y}} \Gamma(y, y') \left\langle w, \phi(x, y') - \phi(x, y)\right\rangle + \Delta(y, y'). \tag{24}$$

Here $\phi(x, y)$ is a *joint* feature map, $\Delta(y, y') \geq 0$ describes the cost of misclassifying y by y', and $\Gamma(y, y') \geq 0$ is a scaling term which indicates by how much the large margin property should be enforced. For instance, Taskar et al. (2004) choose $\Gamma(y, y') = 1$. On the other hand, Tsochantaridis et al. (2005) suggest $\Gamma(y, y') = \Delta(y, y')$, which reportedly yields better performance. Finally, McAllester (2007) recently suggested generic functions $\Gamma(y, y')$.

The logistic loss can also be interpreted as the negative log-likelihood of a conditional exponential family model:

$$p(y|x; w) := \exp(\langle w, \phi(x, y)\rangle - g(w|x)), \tag{25}$$

where the normalizing constant $g(w|x)$, often called the log-partition function, reads

$$g(w|x) := \log \sum_{y' \in \mathcal{Y}} \exp\left(\left\langle w, \phi(x, y')\right\rangle\right).$$

As a consequence of the Hammersley-Clifford theorem (Jordan, 2002) every exponential family distribution corresponds to a undirected graphical model. In our case this implies that the labels y factorize according to an undirected graphical model. A large number of problems have been addressed by this setting, amongst them named entity tagging (Lafferty et al., 2001), sequence alignment (Tsochantaridis et al., 2005), segmentation (Rätsch et al., 2007) and path planning (Ratliff et al., 2006). It is clearly impossible to give examples of all settings in this section, nor would a brief summary do this field any justice. We therefore refer the reader to the edited volume by Bakir et al. (2007) and the references therein.

If the underlying graphical model is tractable then efficient inference algorithms based on dynamic programming can be used to compute (23) and (24). We discuss intractable graphical models in Section A.2.1, and now turn our attention to the derivatives of the above structured losses.

When it comes to computing derivatives of the logistic loss, (23), we have

$$\begin{aligned}\partial_w l(x,y,w) &= \frac{\sum_{y'} \phi(x,y') \exp \langle w, \phi(x,y')\rangle}{\sum_{y'} \exp \langle w, \phi(x,y')\rangle} - \phi(x,y)\\ &= \mathbf{E}_{y' \sim p(y'|x)} \left[\phi(x,y')\right] - \phi(x,y).\end{aligned}$$

where $p(y|x)$ is the exponential family model (25). In the case of (24) we denote by $\bar{y}(x)$ the argmax of the RHS, that is

$$\bar{y}(x) := \operatorname*{argmax}_{y'} \Gamma(y,y') \left\langle w, \phi(x,y') - \phi(x,y) \right\rangle + \Delta(y,y').$$

This allows us to compute the derivative of $l(x,y,w)$ as

$$\partial_w l(x,y,w) = \Gamma(y,\bar{y}(x)) \left[\phi(x,\bar{y}(x)) - \phi(x,y)\right].$$

In the case where the loss is maximized for more than one distinct value $\bar{y}(x)$ we may average over the individual values, since any convex combination of such terms lies in the subdifferential.

Note that (24) majorizes $\Delta(y,y^*)$, where $y^* := \operatorname{argmax}_{y'} \langle w, \phi(x,y')\rangle$ (Tsochantaridis et al., 2005). This can be seen via the following series of inequalities:

$$\Delta(y,y^*) \leq \Gamma(y,y^*) \langle w, \phi(x,y^*) - \phi(x,y)\rangle + \Delta(y,y^*) \leq l(x,y,w).$$

The first inequality follows because $\Gamma(y,y^*) \geq 0$ and y^* maximizes $\langle w, \phi(x,y')\rangle$ thus implying that $\Gamma(y,y^*) \langle w, \phi(x,y^*) - \phi(x,y)\rangle \geq 0$. The second inequality follows by definition of the loss.

We conclude this section with a simple lemma which is at the heart of several derivations of Joachims (2005). While the proof in the original paper is far from trivial, it is straightforward in our setting:

Lemma 8 *Denote by $\delta(y,y')$ a loss and let $\phi(x_i,y_i)$ be a feature map for observations (x_i,y_i) with $1 \leq i \leq m$. Moreover, denote by X,Y the set of all m patterns and labels respectively. Finally let*

$$\Phi(X,Y) := \sum_{i=1}^{m} \phi(x_i,y_i) \text{ and } \Delta(Y,Y') := \sum_{i=1}^{m} \delta(y_i,y'_i).$$

Then the following two losses are equivalent:

$$\sum_{i=1}^{m} \max_{y'} \left\langle w, \phi(x_i,y') - \phi(x_i,y_i)\right\rangle + \delta(y_i,y') \text{ and } \max_{Y'} \left\langle w, \Phi(X,Y') - \Phi(X,Y)\right\rangle + \Delta(Y,Y').$$

This is immediately obvious, since both feature map and loss decompose, which allows us to perform maximization over Y' by maximizing each of its m components. In doing so, we showed that aggregating all data and labels into a single feature map and loss yields results identical to minimizing the sum over all individual losses. This holds, in particular, for the sample error loss of Joachims (2005). Also note that this equivalence does *not* hold whenever $\Gamma(y,y')$ is not constant.

A.2.1 INTRACTABLE MODELS

We now discuss cases where computing $l(x,y,w)$ itself is too expensive. For instance, for intractable graphical models, the computation of $\sum_y \exp \langle w, \phi(x,y)\rangle$ cannot be computed efficiently. Wainwright and Jordan (2003) propose the use of a convex majorization of the log-partition function in those cases. In our setting this means that instead of dealing with

$$l(x,y,w) = g(w|x) - \langle w, \phi(x,y)\rangle \text{ where } g(w|x) := \log \sum_y \exp \langle w, \phi(x,y)\rangle$$

one uses a more easily computable convex upper bound on g via

$$\sup_{\mu \in \mathrm{MARG}(x)} \langle w, \mu\rangle + H_{\mathrm{Gauss}}(\mu|x). \tag{26}$$

Here $\mathrm{MARG}(x)$ is an outer bound on the conditional marginal polytope associated with the map $\phi(x,y)$. Moreover, $H_{\mathrm{Gauss}}(\mu|x)$ is an upper bound on the entropy by using a Gaussian with identical variance. More refined tree decompositions exist, too. The key benefit of our approach is that the solution μ of the optimization problem (26) can immediately be used as a gradient of the upper bound. This is computationally rather efficient.

Likewise, note that Taskar et al. (2004) use relaxations when solving structured estimation problems of the form

$$l(x,y,w) = \max_{y'} \Gamma(y,y') \left\langle w, \phi(x,y') - \phi(x,y)\right\rangle + \Delta(y,y'),$$

by enlarging the domain of maximization with respect to y'. For instance, instead of an integer programming problem we might relax the setting to a linear program which is much cheaper to solve. This, again, provides an upper bound on the original loss function.

In summary, we have demonstrated that convex relaxation strategies are well applicable for bundle methods. In fact, the results of the corresponding optimization procedures can be used directly for further optimization steps.

A.3 Scalar Multivariate Performance Scores

We now discuss a series of structured loss functions and how they can be implemented efficiently. For the sake of completeness, we give a concise representation of previous work on multivariate performance scores and ranking methods. All these loss functions rely on having access to $\langle w, x\rangle$, which can be computed efficiently by using the same operations as in Section A.1.

A.3.1 ROC SCORE

Denote by $f = Xw$ the vector of function values on the training set. It is well known that the area under the ROC curve is given by

$$\mathrm{AUC}(x,y,w) = \frac{1}{m_+ m_-} \sum_{y_i < y_j} \mathbf{I}(\langle w, x_i\rangle < \langle w, x_j\rangle),$$

where m_+ and m_- are the numbers of positive and negative observations respectively, and $\mathbf{I}(\cdot)$ is indicator function. Directly optimizing the cost $1 - \mathrm{AUC}(x,y,w)$ is difficult as it is not continuous

Algorithm 7 ROCScore(X, y, w)

```
1:  input: Feature matrix $X$, labels $y$, and weight vector $w$
2:  initialization: $s_- = m_-$ and $s_+ = 0$ and $l = \mathbf{0}_m$ and $c = Xw - \frac{1}{2}y$
3:  $\pi \leftarrow \{1, \ldots, m\}$ sorted in ascending order of $c$
4:  for $i = 1$ to $m$ do
5:      if $y_{\pi_i} = -1$ then
6:          $l_{\pi_i} \leftarrow s_+$ and $s_- \leftarrow s_- - 1$
7:      else
8:          $l_{\pi_i} \leftarrow -s_-$ and $s_+ \leftarrow s_+ + 1$
9:      end if
10: end for
11: Rescale $l \leftarrow l/(m_+ m_-)$ and compute $r = \langle l, c \rangle$ and $g = l^\top X$.
12: return Risk $r$ and subgradient $g$
```

in w. By using $\max(0, 1 + \langle w, x_i - x_j \rangle)$ as the surrogate loss function for all pairs (i, j) for which $y_i < y_j$ we have the following convex multivariate empirical risk

$$R_{\text{emp}}(w) = \frac{1}{m_+ m_-} \sum_{y_i < y_j} \max(0, 1 + \langle w, x_i - x_j \rangle) = \frac{1}{m_+ m_-} \sum_{y_i < y_j} \max(0, 1 + f_i - f_j). \tag{27}$$

Obviously, we could compute $R_{\text{emp}}(w)$ and its derivative by an $O(m^2)$ operation. However Joachims (2005) shows that both can be computed in $O(m \log m)$ time using a sorting operation, which we now describe.

Denote by $c = f - \frac{1}{2}y$ an auxiliary variable and let i and j be indices such that $y_i = -1$ and $y_j = 1$. It follows that $c_i - c_j = 1 + f_i - f_j$. The efficient algorithm is due to the observation that there are at most m distinct terms $c_k,\ k = 1, \ldots, m$, each with different frequency l_k and sign, appear in (27). These frequencies l_k can be determined by first sorting c in ascending order then scanning through the labels according to the sorted order of c and keeping running statistics such as the number s_- of negative labels yet to encounter, and the number s_+ of positive labels encountered. When visiting y_k, we know c_k should appears s_+ (or s_-) times with positive (or negative) sign in (27) if $y_k = -1$ (or $y_k = 1$). Algorithm 7 spells out explicitly how to compute $R_{\text{emp}}(w)$ and its subgradient.

A.3.2 ORDINAL REGRESSION

Essentially the same preference relationships need to hold for ordinal regression. The only difference is that y_i need not take on binary values any more. Instead, we may have an arbitrary number of different values y_i (e.g., 1 corresponding to 'strong reject' up to 10 corresponding to 'strong accept', when it comes to ranking papers for a conference). That is, we now have $y_i \in \{1, \ldots, n\}$ rather than $y_i \in \{\pm 1\}$. Our goal is to find some w such that $\langle w, x_i - x_j \rangle < 0$ whenever $y_i < y_j$. Whenever this relationship is not satisfied, we incur a cost $C(y_i, y_j)$ for preferring x_i to x_j. For examples, $C(y_i, y_j)$ could be constant, that is, $C(y_i, y_j) = 1$ (Joachims, 2006) or linear, that is, $C(y_i, y_j) = y_j - y_i$.

Denote by m_i the number of x_j for which $y_j = i$. In this case, there are $\bar{M} = m^2 - \sum_{i=1}^n m_i^2$ pairs (y_i, y_j) for which $y_i \neq y_j$; this implies that there are $M = \bar{M}/2$ pairs (y_i, y_j) such that $y_i < y_j$.

Normalizing by the total number of comparisons we may write the overall cost of the estimator as

$$\frac{1}{M}\sum_{y_i<y_j} C(y_i,y_j)\mathbf{I}(\langle w,x_i\rangle > \langle w,x_j\rangle) \text{ where } M=\frac{1}{2}\left[m^2-\sum_i^n m_i^2\right].$$

Using the same convex majorization as above when we were maximizing the ROC score, we obtain an empirical risk of the form

$$R_{\text{emp}}(w)=\frac{1}{M}\sum_{y_i<y_j} C(y_i,y_j)\max(0,1+\langle w,x_i-x_j\rangle).$$

Now the goal is to find an efficient algorithm for obtaining the number of times when the individual losses are nonzero such as to compute both the value and the gradient of $R_{\text{emp}}(w)$. The complication arises from the fact that observations x_i with label y_i may appear in either side of the inequality depending on whether $y_j<y_i$ or $y_j>y_i$. This problem can be solved as follows: sort $f=Xw$ in ascending order and traverse it while keeping track of how many items with a lower value y_j are no more than 1 apart in terms of their value of f_i. This way we may compute the count statistics efficiently. Algorithm 8 describes the details, generalizing the results of Joachims (2006). Again, its runtime is $O(m\log m)$, thus allowing for efficient computation.

A.3.3 PREFERENCE RELATIONS

In general, our loss may be described by means of a set of preference relations $j\succeq i$ for arbitrary pairs $(i,j)\in\{1,\dots m\}^2$ associated with a cost $C(i,j)$ which is incurred whenever i is ranked above j. This set of preferences may or may not form a partial or a total order on the domain of all observations. In these cases efficient computations along the lines of Algorithm 8 exist. In general, this is not the case and we need to rely on the fact that the set P containing all preferences is sufficiently small that it can be enumerated efficiently. The risk is then given by

$$\frac{1}{|P|}\sum_{(i,j)\in P} C(i,j)\mathbf{I}(\langle w,x_i\rangle > \langle w,x_j\rangle).$$

Again, the same majorization argument as before allows us to write a convex upper bound

$$R_{\text{emp}}(w)=\frac{1}{|P|}\sum_{(i,j)\in P} C(i,j)\max\left(0,1+\langle w,x_i\rangle-\langle w,x_j\rangle\right)$$

$$\text{where } \partial_w R_{\text{emp}}(w)=\frac{1}{|P|}\sum_{(i,j)\in P} C(i,j)\begin{cases}0 & \text{if } \langle w,x_j-x_i\rangle\geq 1\\ x_i-x_j & \text{otherwise.}\end{cases}$$

The implementation is straightforward, as given in Algorithm 9.

A.3.4 RANKING

In webpage and document ranking we are often in a situation similar to that described in Section A.3.2, however with the difference that we do not only care about objects x_i being ranked according to scores y_i but moreover that different degrees of importance are placed on different documents.

Algorithm 8 OrdinalRegression(X, y, w, C)

```
1:  input: Feature matrix $X$, labels $y$, weight vector $w$, and score matrix $C$
2:  initialization: $l = \mathbf{0}_n$ and $u_i = m_i \, \forall i \in [n]$ and $r = 0$ and $g = \mathbf{0}_m$
3:  Compute $f = Xw$ and set $c = [f - \frac{1}{2}, f + \frac{1}{2}] \in \mathbb{R}^{2m}$ (concatenate the vectors)
4:  Compute $M = (m^2 - \sum_{i=1}^{n} m_i^2)/2$
5:  Rescale $C \leftarrow C/M$
6:  $\pi \leftarrow \{1, \ldots, 2m\}$ sorted in ascending order of $c$
7:  for $i = 1$ to $2m$ do
8:      $j = \pi_i \bmod m$
9:      if $\pi_i \leq m$ then
10:         for $k = 1$ to $y_j - 1$ do
11:             $r \leftarrow r - C(k, y_j) u_k c_j$
12:             $g_j \leftarrow g_j - C(k, y_j) u_k$
13:         end for
14:         $l_{y_j} \leftarrow l_{y_j} + 1$
15:     else
16:         for $k = y_j + 1$ to $n$ do
17:             $r \leftarrow r + C(y_j, k) l_k c_{j+m}$
18:             $g_j \leftarrow g_j + C(y_j, k) l_k$
19:         end for
20:         $u_{y_j} \leftarrow u_{y_j} - 1$
21:     end if
22: end for
23: $g \leftarrow g^\top X$
24: return: Risk $r$ and subgradient $g$
```

Algorithm 9 Preference(X, w, C, P)

```
1:  input: Feature matrix $X$, weight vector $w$, score matrix $C$, and preference set $P$
2:  initialization: $r = 0$ and $g = \mathbf{0}_m$
3:  Compute $f = Xw$
4:  while $(i, j) \in P$ do
5:      if $f_j - f_i < 1$ then
6:          $r \leftarrow r + C(i, j)(1 + f_i - f_j)$
7:          $g_i \leftarrow g_i + C(i, j)$ and $g_j \leftarrow g_j - C(i, j)$
8:      end if
9:  end while
10: $g \leftarrow g^\top X$
11: return Risk $r$ and subgradient $g$
```

The information retrieval literature is full with a large number of different scoring functions. Examples are criteria such as *Normalized Discounted Cumulative Gain (NDCG)*, *Mean Reciprocal Rank (MRR)*, *Precision@n*, or *Expected Rank Utility (ERU)*. They are used to address the issue of evaluating rankers, search engines or recommender sytems (Voorhees, 2001; Jarvelin and Kekalainen, 2002; Breese et al., 1998; Basilico and Hofmann, 2004). For instance, in webpage

Algorithm 10 Ranking(X, y, w)

```
1: input: Feature matrix $X$, relevances $y$, and weight vector $w$
2: Compute vectors $a$ and $b(y)$ according to some ranking measure
3: Compute $f = Xw$
4: Compute elements of matrix $C_{ij} = c_i f_j - b_i a_j$
5: $\pi = \text{LinearAssignment}(C)$
6: $r = c^\top (f(\pi) - f) + (a - a(\pi))^\top b$
7: $g = c(\pi^{-1}) - c$ and $g \leftarrow g^\top X$
8: return Risk $r$ and subgradient $g$
```

ranking only the first k retrieved documents that matter, since users are unlikely to look beyond the first k, say 10, retrieved webpages in an internet search. Le and Smola (2007) show that these scores can be optimized directly by minimizing the following loss:

$$l(X, y, w) = \max_{\pi} \sum_i c_i \left\langle w, x_{\pi(i)} - x_i \right\rangle + \langle a - a(\pi), b(y) \rangle \,. \tag{28}$$

Here c_i is a monotonically decreasing sequence, the documents are assumed to be arranged in order of decreasing relevance, π is a permutation, the vectors a and $b(y)$ depend on the choice of a particular ranking measure, and $a(\pi)$ denotes the permutation of a according to π. Pre-computing $f = Xw$ we may rewrite (28) as

$$l(f, y) = \max_{\pi} \left[c^\top f(\pi) - a(\pi)^\top b(y) \right] - c^\top f + a^\top b(y)$$

and consequently the derivative of $l(X, y, w)$ with respect to w is given by

$$\partial_w l(X, y, w) = (c(\bar{\pi}^{-1}) - c)^\top X \text{ where } \bar{\pi} = \operatorname*{argmax}_{\pi} c^\top f(\pi) - a(\pi)^\top b(y).$$

Here π^{-1} denotes the inverse permutation, such that $\pi \circ \pi^{-1} = 1$. Finding the permutation maximizing $c^\top f(\pi) - a(\pi)^\top b(y)$ is a linear assignment problem which can be easily solved by the Hungarian Marriage algorithm, that is, the Kuhn-Munkres algorithm.

The original papers by Kuhn (1955) and Munkres (1957) implied an algorithm with $O(m^3)$ cost in the number of terms. Later, Karp (1980) suggests an algorithm with expected quadratic time in the size of the assignment problem (ignoring log-factors). Finally, Orlin and Lee (1993) propose a linear time algorithm for large problems. Since in our case the number of pages is fairly small (in the order of 50 to 200 *per query*) the scaling behavior per query is not too important. We used an existing implementation due to Jonker and Volgenant (1987).

Note also that training sets consist of a *collection* of ranking problems, that is, we have several ranking problems of size 50 to 200. By means of parallelization we are able to distribute the work onto a cluster of workstations, which is able to overcome the issue of the rather costly computation per collection of queries. Algorithm 10 spells out the steps in detail.

A.3.5 CONTINGENCY TABLE SCORES

Joachims (2005) observed that F_β scores and related quantities dependent on a contingency table can also be computed efficiently by means of structured estimation. Such scores depend in general on the number of true and false positives and negatives alike. Algorithm 11 shows how a corresponding empirical risk and subgradient can be computed efficiently. As with the previous losses, here again we use convex majorization to obtain a tractable optimization problem.

Given a set of labels y and an estimate y', the numbers of true positives (T_+), true negatives (T_-), false positives (F_+), and false negatives (F_-) are determined according to a contingency table as follows:

	$y > 0$	$y < 0$
$y' > 0$	T_+	F_+
$y' < 0$	F_-	T_-

In the sequel, we denote by $m_+ = T_+ + F_-$ and $m_- = T_- + F_+$ the numbers of positives and negative labels in y, respectively. We note that F_β score can be computed based on the contingency table (Joachims, 2005) as

$$F_\beta(T_+, T_-) = \frac{(1+\beta^2)T_+}{T_+ + m_- - T_- + \beta^2 m_+}.$$

If we want to use $\langle w, x_i \rangle$ to estimate the label of observation x_i, we may use the following structured loss to "directly" optimize w.r.t. F_β score (Joachims, 2005):

$$l(X, y, w) = \max_{y'} \left[(y' - y)^\top f + \Delta(T_+, T_-) \right],$$

where $f = Xw$, $\Delta(T_+, T_-) := 1 - F_\beta(T_+, T_-)$, and (T_+, T_-) is determined by using y and y'. Since Δ does not depend on the specific choice of (y, y') but rather just on which sets they disagree, l can be maximized as follows: Enumerating all possible $m_+ m_-$ contingency tables in a way such that given a configuration (T_+, T_-), T_+ (T_-) positive (negative) observations x_i with largest (lowest) value of $\langle w, x_i \rangle$ are labeled as positive (negative). This is effectively implemented as a nested loop hence run in $O(m^2)$ time. Algorithm 11 describes the procedure in details.

A.4 Vector Loss Functions

Next we discuss "vector" loss functions, that is, functions where w is best described as a matrix (denoted by W) and the loss depends on Wx. Here, we have feature vector $x \in \mathbb{R}^d$, label $y \in \mathbb{R}^k$, and weight matrix $W \in \mathbb{R}^{d \times k}$. We also denote feature matrix $X \in \mathbb{R}^{m \times d}$ as a matrix of m feature vectors x_i, and stack up the columns W_i of W as a vector w.

Some of the most relevant cases are multiclass classification using both the exponential families model and structured estimation, hierarchical models, that is, ontologies, and multivariate regression. Many of those cases are summarized in Table 6.

A.4.1 UNSTRUCTURED SETTING

The simplest loss is multivariate regression, where $l(x, y, W) = \frac{1}{2}(y - x^\top W)^\top M (y - x^\top W)$. In this case it is clear that by pre-computing XW subsequent calculations of the loss and its gradient are significantly accelerated.

A second class of important losses is given by plain multiclass classification problems, for example, recognizing digits of a postal code or categorizing high-level document categories. In this case, $\phi(x, y)$ is best represented by $e_y \otimes x$ (using a linear model). Clearly we may view $\langle w, \phi(x, y) \rangle$

Algorithm 11 $\mathrm{F}_\beta(X, y, w)$

```
1: input: Feature matrix $X$, labels $y$, and weight vector $w$
2: Compute $f = Xw$
3: $\pi^+ \leftarrow \{i : y_i = 1\}$ sorted in descending order of $f$
4: $\pi^- \leftarrow \{i : y_i = -1\}$ sorted in ascending order of $f$
5: Let $p_0 = 0$ and $p_i = 2\sum_{k=i}^{m_+} f_{\pi_k^+},\ i = 1, \dots, m_+$
6: Let $n_0 = 0$ and $n_i = 2\sum_{k=i}^{m_-} f_{\pi_k^-},\ i = 1, \dots, m_-$
7: $y' \leftarrow -y$ and $r \leftarrow -\infty$
8: for $i = 0$ to $m_+$ do
9:    for $j = 0$ to $m_-$ do
10:       $r_{\mathrm{tmp}} = \Delta(i, j) - p_i + n_j$
11:       if $r_{\mathrm{tmp}} > r$ then
12:          $r \leftarrow r_{\mathrm{tmp}}$
13:          $T_+ \leftarrow i$ and $T_- \leftarrow j$
14:       end if
15:    end for
16: end for
17: $y'_{\pi_i^+} \leftarrow 1,\ i = 1, \dots, T_+$
18: $y'_{\pi_i^-} \leftarrow -1,\ i = 1, \dots, T_-$
19: $g \leftarrow (y' - y)^\top X$
20: return Risk $r$ and subgradient $g$
```

as an operation which chooses a column indexed by y from xW, since all labels y correspond to a different weight vector W_y. Formally we set $\langle w, \phi(x, y)\rangle = [xW]_y$. In this case, structured estimation losses can be rewritten as

$$l(x, y, W) = \max_{y'} \Gamma(y, y') \left\langle W_{y'} - W_y, x\right\rangle + \Delta(y, y') \tag{29}$$
$$\text{and } \partial_W l(x, y, W) = \Gamma(y, y^*)(e_{y^*} - e_y) \otimes x.$$

Here Γ and Δ are defined as in Section A.2 and y^* denotes the value of y' for which the RHS of (29) is maximized. This means that for unstructured multiclass settings we may simply compute xW. Since this needs to be performed for all observations x_i we may take advantage of fast linear algebra routines and compute $f = XW$ for efficiency. Likewise note that computing the gradient over m observations is now a matrix-matrix multiplication, too: denote by G the matrix of rows of gradients $\Gamma(y_i, y_i^*)(e_{y_i^*} - e_{y_i})$. Then $\partial_W R_{\mathrm{emp}}(X, y, W) = G^\top X$. Note that G is very sparse with at most two nonzero entries per row, which makes the computation of $G^\top X$ essentially as expensive as two matrix vector multiplications. Whenever we have many classes, this may yield significant computational gains.

Log-likelihood scores of exponential families share similar expansions. We have

$$l(x, y, W) = \log \sum_{y'} \exp \left\langle w, \phi(x, y')\right\rangle - \langle w, \phi(x, y)\rangle = \log \sum_{y'} \exp \left\langle W_{y'}, x\right\rangle - \langle W_y, x\rangle$$
$$\partial_W l(x, y, W) = \frac{\sum_{y'} (e_{y'} \otimes x) \exp \left\langle W_{y'}, x\right\rangle}{\sum_{y'} \exp \left\langle W_{y'}, x\right\rangle} - e_y \otimes x.$$

The main difference to the soft-margin setting is that the gradients are *not* sparse in the number of classes. This means that the computation of gradients is slightly more costly.

A.4.2 ONTOLOGIES

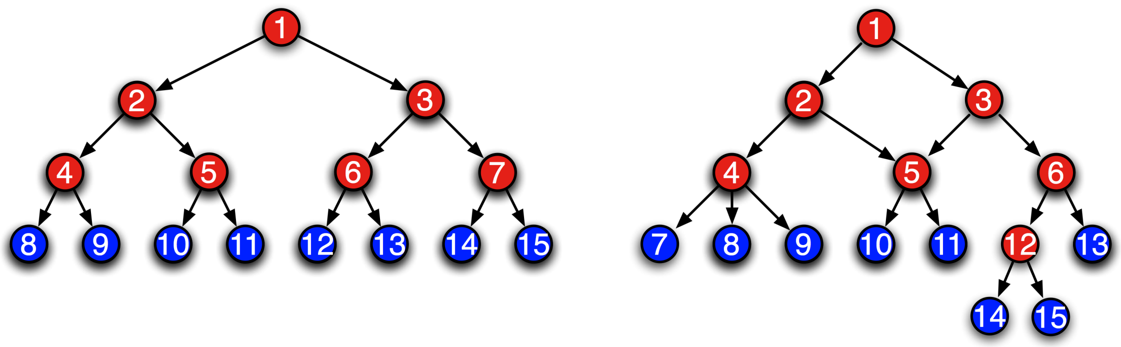


Figure 13: Two ontologies. **Left:** a binary hierarchy with internal nodes $\{1, \ldots, 7\}$ and labels $\{8, \ldots 15\}$. **Right:** a generic directed acyclic graph with internal nodes $\{1, \ldots, 6, 12\}$ and labels $\{7, \ldots, 11, 13, \ldots, 15\}$. Note that node 5 has two parents, namely nodes 2 and 3. Moreover, the labels need not be found at the same level of the tree: nodes 14 and 15 are one level lower than the rest of the nodes.

Assume that the labels we want to estimate can be found to belong to a directed acyclic graph (DAG). For instance, this may be a gene-ontology graph (Ashburner et al., 2000) a patent hierarchy (Cai and Hofmann, 2004), or a genealogy. In these cases we have a hierarchy of categories to which an element x may belong. Figure 13 gives two examples of such directed acyclic graphs. The first example is a binary tree, while the second contains nodes with different numbers of children (e.g., node 4 and 12), nodes at different levels having children (e.g., nodes 5 and 12), and nodes which have more than one parent (e.g., node 5). It is a well known fundamental property of trees that they have at most as many internal nodes as they have leaf nodes.

It is now our goal to build a classifier which is able to categorize observations according to which leaf node they belong to (each leaf node is assigned a label y). Denote by $k+1$ the number of nodes in the DAG including the root node. In this case we may design a feature map $\phi(y) \in \mathbb{R}^k$ (Cai and Hofmann, 2004) by associating with every label y the vector describing the path from the root node to y, ignoring the root node itself. For instance, for the first DAG in Figure 13 we have

$$\phi(8) = (1,0,1,0,0,0,1,0,0,0,0,0,0,0) \text{ and } \phi(13) = (0,1,0,0,1,0,0,0,0,0,0,1,0,0)$$

Whenever several paths are admissible, as in the right DAG of Figure 13 we average over all possible paths. For example, we have

$$\phi(10) = (0.5,0.5,0,1,0,0,0,0,1,0,0,0,0,0) \text{ and } \phi(15) = (0,1,0,0,1,0,0,0,0,0,0,1,0,0,1).$$

Also note that the lengths of the paths need not be the same (e.g., to reach 15 it takes a longer path than to reach 13). Likewise, it is natural to assume that $\Delta(y, y')$, that is, the cost for mislabeling y

Algorithm 12 Ontology(X, y, W)

```
1: input: Feature matrix $X \in \mathbb{R}^{m \times d}$, labels $y$, and weight matrix $W \in \mathbb{R}^{d \times k}$
2: initialization: $G = \mathbf{0} \in \mathbb{R}^{m \times k}$ and $r = 0$
3: Compute $f = XW$ and let $f_i = x_i W$
4: for $i = 1$ to $m$ do
5:     Let $D_i$ be the DAG with edges annotated with the values of $f_i$
6:     Traverse $D_i$ to find a path $y^*$ that maximizes the value $z_{y^*} := \sum_{j=1}^{k} [\phi(y^*)]_j f_{ij} + \Delta(y_i, y^*)$
7:     $G_i = \phi(y^*) - \phi(y_i)$
8:     $r \leftarrow r + z_{y^*} - z_{y_i}$
9: end for
10: $g = G^\top X$
11: return Risk $r$ and subgradient $g$
```

as y' will depend on the similarity of the path. In other words, it is likely that the cost for placing x into the wrong sub-sub-category is less than getting the main category of the object wrong.

To complete the setting, note that for $\phi(x, y) = \phi(y) \otimes x$ the cost of computing all labels is k inner products, since the value of $\langle w, \phi(x, y) \rangle$ for a particular y can be obtained by the sum of the contributions for the segments of the path. This means that the values for *all* terms can be computed by a simple breadth first traversal through the graph. As before, we may make use of vectorization in our approach, since we may compute $xW \in \mathbb{R}^k$ to obtain the contributions on all segments of the DAG before performing the graph traversal. Since we have m patterns x_i we may vectorize matters by pre-computing XW.

Also note that $\phi(y) - \phi(y')$ is nonzero only for those edges where the paths for y and y' differ. Hence we only change weights on those parts of the graph where the categorization differs. Algorithm 12 describes the subgradient and loss computation for the soft-margin type of loss function.

The same reasoning applies to estimation when using an exponential families model. The only difference is that we need to compute a *soft-max* over paths rather than exclusively choosing the best path over the ontology. Again, a breadth-first recursion suffices: each of the leaves y of the DAG is associated with a probability $p(y|x)$. To obtain $\mathbf{E}_{y \sim p(y|x)} [\phi(y)]$ all we need to do is perform a bottom-up traversal of the DAG summing over all probability weights on the path. Wherever a node has more than one parent, we distribute the probability weight equally over its parents.

Appendix B. Proofs

This section contains the proofs of Theorems 4, 5, and 7, along with the technical lemmas required for these.

B.1 Proof of Theorem 4

To show Theorem 4 we need several technical intermediate steps. Let $\gamma_t := J(w_t) - J_t(w_t)$ and recall that $\varepsilon_t := \min_{t' \le t} J(w_{t'}) - J_t(w_t)$. The following lemma establishes some useful properties of γ_t and ε_t.

Lemma 9 *We have $J_{t'}(w_{t'}) \leq J_t(w_t) \leq J(w^*) \leq J(w_t) = J_{t+1}(w_t)$ for all $t' \leq t$. Furthermore, ϵ_t is monotonically decreasing with $\epsilon_t - \epsilon_{t+1} \geq J_{t+1}(w_{t+1}) - J_t(w_t) \geq 0$. Also, ϵ_t upper bounds the distance from optimality via $\gamma_t \geq \epsilon_t \geq \min_{t' \leq t} J(w_{t'}) - J(w^*)$.*

Proof Since $J_{t'}(w) \leq J_t(w) \leq J(w)$ for all $t' \leq t$ this property also applies to their respective minima. Moreover, since w^* minimizes $J(w)$ we have $J(w^*) \leq J(w_t)$. Since Taylor expansions are exact at the point of expansion $J(w_t) = J_{t+1}(w_t)$. The first inequality follows from the definition of ϵ_t, and the fact that J_t is monotonically increasing. Finally, since $J_{t'+1}(w_{t'}) = J(w_{t'})$ it is easy to see that $\gamma_t \geq \epsilon_t = \min_{t' \leq t} J(w_{t'}) - J_t(w_t) \geq \min_{t' \leq t} J(w_{t'}) - J(w^*)$. ■

Our second technical lemma allows us to bound the maximum value of a concave function provided that we know its first derivative and a bound on the second derivative.

Lemma 10 *Denote by $f : [0,1] \to \mathbb{R}$ a concave function with $f(0) = 0$, $f'(0) = l$, and $|f''(x)| \leq K \; \forall x \in [0,1]$. Then we have $\max_{x \in [0,1]} f(x) \geq \frac{l}{2} \min(\frac{l}{K}, 1)$.*

Proof We first observe that $g(x) := lx - \frac{K}{2}x^2 \leq f(x) \; \forall x$ implies $\max_{x \in [0,1]} f(x) \geq \max_{x \in [0,1]} g(x)$. g attains the unconstrained maximum $\frac{l^2}{2K}$ at $x = \frac{l}{K}$. Since g is monotonically increasing in $[0, \frac{l}{K}]$, if $l > K$ we pick $x = 1$ which yields constrained maximum $l - \frac{K}{2} > \frac{l}{2}$. Taking the minimum over both maxima proves the claim.

■

To apply the above result, we need to compute the gradient and Hessian of $J^*_{t+1}(\alpha)$ with respect to the search direction $((1-\eta)\alpha_t, \eta)$. The following lemma takes care of the gradient:

Lemma 11 *Denote by α_t the solution of (9) at time instance t. Moreover, denote by $\bar{A} = [A, a_{t+1}]$ and $\bar{b} = [b, b_{t+1}]$ the extended matrices and vectors needed to define the dual problem for step $t+1$, and let $\bar{\alpha} \in \mathbb{R}^{t+1}$. Then the following holds:*

$$\partial_{\bar{\alpha}} J^*_{t+1}([\alpha_t, 0]) = \bar{A}^\top w_t + \bar{b} \text{ and}$$
$$[-\alpha_t, 1]^\top \left[\bar{A}^\top w_t + \bar{b}\right] = J_{t+1}(w_t) - J_t(w_t) = \gamma_t. \tag{30}$$

Proof By the dual connection $\partial\Omega^*(-\lambda^{-1}A\alpha_t) = w_t$. Hence we have that $\partial_{\bar{\alpha}} - \lambda\Omega^*(-\lambda^{-1}\bar{A}\bar{\alpha}) + \bar{\alpha}^\top \bar{b} = \bar{A}^\top w_t + \bar{b}$ for $\bar{\alpha} = [\alpha_t, 0]^\top$. This proves the first claim. To see the second part we eliminate ξ from of the Lagrangian (11) and write the partial Lagrangian

$$L(w, \alpha) = \lambda\Omega(w) + \alpha^\top \left(A^\top w + b\right) \text{ with } \alpha \geq 0.$$

The result follows by noting that at optimality $L(w_t, \alpha_t) = J_t(w_t)$ and $J_{t+1}(w_t) = \lambda\Omega(w_t) + \langle w_t, a_{t+1}\rangle + b_{t+1}$. Consequently we have

$$J_{t+1}(w_t) - J_t(w_t) = \lambda\Omega(w_t) + \langle w_t, a_{t+1}\rangle + b_{t+1} - \lambda\Omega(w_t) - \alpha_t(A^\top w_t + b_t).$$

Rearranging terms proves the claim. ■

To apply Lemma 10 we also need to bound the second derivative.

Lemma 12 *Under the assumptions of Lemma 11 we have*

$$\partial^2_{\bar\alpha} J^*_{t+1}(\bar\alpha) = -\lambda^{-1}\bar{A}^\top \partial^2\Omega^*(-\lambda^{-1}\bar{A}\bar\alpha)\bar{A} \tag{31}$$

$$\text{moreover } \bar{A}[-\alpha_t, 1] = s_t \in \partial_w J(w_t). \tag{32}$$

Proof The first equality is immediate from the chain rule. Next note that $\partial_w \Omega(w_t) = -\lambda^{-1} A\alpha_t$ by dual connection. Since $a_{t+1} \in \partial_w R_{\text{emp}}(w_t)$ the claim follows from $J(w) = R_{\text{emp}}(w) + \lambda\Omega(w)$. ■

This result allows us to express the second derivative of the dual objective function (10) in terms of the gradient of the risk functional. The idea is that as we approach optimality, the second derivative will vanish. We will use this fact to argue that for continuously differentiable losses $R_{\text{emp}}(w)$ we enjoy linear convergence throughout.

Proof [Theorem 4] We overload the notation for J^*_{t+1} by defining the following one dimensional concave function

$$J^*_{t+1}(\eta) := J^*_{t+1}([(1-\eta)\alpha_t, \eta]) = -\lambda\Omega^*(-\lambda^{-1}\bar{A}[(1-\eta)\alpha_t^\top, \eta]) + [(1-\eta)\alpha_t^\top, \eta]\bar{b}.$$

Clearly, $J^*_{t+1}(0) = J_t(w_t)$. Furthermore, by (30), (31), and (32) it follows that

$$\begin{aligned}
\partial_\eta J^*_{t+1}(\eta)|_{\eta=0} &= [-\alpha_t, 1]^\top \partial_{\bar\alpha} J^*_{t+1}([\alpha_t, 0]) = \gamma_t \text{ and} \\
\partial^2_\eta J^*_{t+1}(\eta) &= -\lambda^{-1}[-\alpha_t, 1]^\top \bar{A}^\top \partial^2\Omega^*(-\lambda^{-1}\bar{A}[(1-\eta)\alpha_t, \eta])\bar{A}[-\alpha_t, 1]^\top \\
&= -\lambda^{-1} s_t^\top \partial^2\Omega^*(-\lambda^{-1}\bar{A}[(1-\eta)\alpha_t, \eta]) s_t := r.
\end{aligned}$$

By our assumption on $\|\partial^2\Omega^*\| \le H^*$ we have

$$|r| \le H^* \|s_t\|^2/\lambda.$$

Next we need to bound the gradient of J. For this purpose note that $\partial_w \lambda\Omega(w_t) = -A^\top\alpha_t$ and moreover that $\|\alpha_t\|_1 = 1$. This implies that $\partial_w\lambda\Omega(w_t)$ lies in the convex hull of the past gradients, $a_{t'}$. By our assumption that $\max_{u\in\partial_w R_{\text{emp}}(w)} \|u\| \le G$ it follows that $\|\partial_w\lambda\Omega(w_t)\| \le G$. We conclude that

$$\|s_t\|^2 \le 4G^2 \text{ and } |r| \le 4G^2H^*/\lambda.$$

Invoking Lemma 10 on $J^*_{t+1}(\eta) - J_t(w_t)$ shows that

$$J^*_{t+1}(\eta) - J_t(w_t) \ge \tfrac{\gamma_t}{2}\min(1, \lambda\gamma_t/4G^2H^*).$$

We now upper bound the LHS of the above inequality as follows:

$$\epsilon_t - \epsilon_{t+1} \ge J_{t+1}(w_{t+1}) - J_t(w_t) \ge J^*_{t+1}(\eta) - J_t(w_t) \ge \tfrac{\gamma_t}{2}\min(1, \lambda\gamma_t/4G^2H^*). \tag{33}$$

The first inequality follows from Lemma 9 while the second follows by observing that $J_{t+1}(w_{t+1}) = J^*_{t+1}(\alpha_{t+1}) \ge J^*_{t+1}(\eta)$. The RHS of the third inequality on the other hand can be lower bounded by observing that $\gamma_t \ge \epsilon_t$, which follows from Lemma 9. This in turn obtains (12).

For the second part note that (12) already yields the $\epsilon_t/2$ decrease when $\epsilon_t \ge 4G^2H^*/\lambda$. To show the other parts we need to show that the gradient of the regularized risk vanishes as we converge to the optimal solution. Towards this end, we apply Lemma 10 in the *primal*.[23] This allows us to bound $\|\partial_w J(w_t)\|$ in terms of γ_t. Plugging in the first and second derivative of $J(w_t)$ we obtain

$$\gamma_t \geq \frac{1}{2}\|\partial_w J(w_t)\| \min(1, \|\partial_w J(w_t)\|/H).$$

If $\|\partial_w J(w_t)\| > H$, then $\gamma_t \geq \frac{1}{2}\|\partial_w J(w_t)\|$ which in turn yields $|r| \leq 4\gamma_t^2 H^*/\lambda$. Plugging this into Lemma 10 yields a lower bound on the improvement of $\lambda/8H^*$.

Finally, for $\|\partial_w J(w_t)\| \leq H$ we have $\gamma_t \geq \|\partial_w J(w_t)\|^2/2H$, which implies $|r| \leq 2HH^*\gamma_t/\lambda$. Plugging this into Lemma 10 yields an improvement of $\lambda\gamma_t/4HH^* \geq \lambda\varepsilon_t/4HH^*$.

Since both cases cover the remaining range of convergence, the minimum $\min(\lambda/8H^*, \lambda\varepsilon_t/4HH^*)$ provides a lower bound for the improvement. The crossover point between both terms occurs at $\varepsilon_t = H/2$. Rearranging the conditions leads to the (pessimistic) improvement guarantees of the second claim.

■

Note that a key step in the above analysis involved bounding $r := \partial_\eta^2 J_{t+1}^*(\eta)$. For a number of regularizers tighter bounds can be obtained. The following bounds are essentially due to Shalev-Shwartz and Singer (2006):

- For squared norm regularization, that is, $\Omega^*(\mu) = \frac{1}{2}\|\mu\|_2^2$ we have $r = \|\partial_w J(w_t)\|_2^2$.
- For L_p norm regularization, that is, $\Omega^*(\mu) = \frac{1}{2}\|\mu\|_q^2$ we have $r \leq (q-1)\|\partial_w J(w_t)\|_q^2$.
- For quadratic form regularization with PD matrix B, that is, $\Omega^*(\mu) = \frac{1}{2}\mu B^{-1}\mu$, we have $r = \partial_w J(w_t)^\top B^{-1}\partial_w J(w_t)$.
- For unnormalized entropic regularization we have $\partial_\mu^2\Omega^*(\mu) = \operatorname{diag}\left(e^{\mu^{(1)}}, \ldots, e^{\mu^{(d)}}\right)$. Hence we may bound $r \leq \|\partial_w J(w_t)\|_2^2 \exp(\|\mu\|_\infty)$. Clearly this bound may be very loose whenever μ has only very few large coefficients.
- For normalized entropy regularization, that is, $\Omega^*(\mu) = \log\sum_i \exp\mu^{(i)}$ we have $r \leq \|\partial_w J(w_t)\|_\infty^2$.

B.2 Proof of Theorem 5

We need the following technical lemma for the proof:

Lemma 13 *Let $\langle\rho_1, \rho_2, \ldots\rangle$ be a sequence of non-negative numbers satisfying the following recurrence, for $t \geq 1$: $\rho_t - \rho_{t+1} \geq c(\rho_t)^2$, where $c > 0$ is a constant. Then for all integers $t \geq 1$,*

$$\rho_t \leq \frac{1}{c(t - 1 + \frac{1}{\rho_1 c})}.$$

Furthermore $\rho_t \leq \rho$ whenever

$$t \geq \frac{1}{c\rho} - \frac{1}{\rho_1 c} + 1.$$

23. Define $\bar{J}(\eta) := J(w_t) - J(w_t + \eta p)$ where $p = -\frac{\partial_w J(w_t)}{\|\partial_w J(w_t)\|}$ is the unit-length gradient. We see that $\frac{d}{d\eta}\bar{J}(\eta)\Big|_{\eta=0} = [-\partial_w J(w_t + \eta p)^\top p]|_{\eta=0} = \|\partial_w J(w_t)\|$, and $\bar{J}(0) = 0$. Hence Lemma 10 is applicable in this case.

This is Sublemma 5.4 of Abe et al. (2001) which is easily proven by induction. Now we can prove the main result.

Proof [Theorem 5] For any $\varepsilon_t > 4G^2H^*/\lambda$ it follows from (12) that $\varepsilon_{t+1} \leq \varepsilon_t/2$. Moreover, $\varepsilon_0 \leq J(0)$, since we know that J is non-negative. Hence we need at most $\log_2[\lambda J(0)/4G^2H^*]$ to achieve this level of precision. Subsequently we have

$$\varepsilon_t - \varepsilon_{t+1} \geq \frac{\lambda}{8G^2H^*}\varepsilon_t^2.$$

Invoking Lemma 13 by setting $c = \frac{\lambda}{8G^2H^*}$ and $\rho_1 = 4G^2H^*/\lambda$ shows that $\varepsilon_t \leq \varepsilon$ after at most $\frac{8G^2H^*}{\lambda\varepsilon} - 1$ more steps. This proves the first claim.

To analyze convergence in the second case we need to study two additional phases: for $\varepsilon_t \in [H/2, 4G^2H^*/\lambda]$ we see constant progress. Hence it takes us $4\lambda^{-2}[8G^2(H^*)^2 - HH^*\lambda]$ steps to cover this interval. Finally in the third phase we have $\varepsilon_{t+1} \leq \varepsilon_t[1-\lambda/4HH^*]$. Starting from $\varepsilon_t = H/2$ we need $\log_2[2\varepsilon/H]/\log_2[1-\lambda/4HH^*]$ steps to converge. Expanding the logarithm in the denominator close to 1 proves the claim. ■

B.3 Proof of Theorem 7

We first note that the termination criterion of Algorithm 3 is slightly different from that of Algorithm 2. In order to apply the convergence results for Algorithm 2 to Algorithm 3 we redefine the following notations:

$$\begin{aligned} \varepsilon_t &:= J(w_t^b) - J_t(w_t) \\ a_{t+1} &\in \partial_w R_{\text{emp}}(w_t^c), \\ b_{t+1} &:= R_{\text{emp}}(w_t^c) - \langle w_t, a_{t+1}\rangle, \end{aligned} \tag{34}$$

where

$$\begin{aligned} \eta_t &:= \operatorname*{argmin}_{\eta} J(w_{t-1}^b + \eta(w_t - w_{t-1}^b)), \\ w_t^b &:= \hat{w}_{t-1} + \eta_t(\bar{w}_t - \hat{w}_{t-1}), \quad \text{and} \\ w_t^c &:= (1-\theta)w_t^b + \theta w_t. \end{aligned}$$

Then we state and prove the following lemma which is crucial to the application of Lemma 11 in the proof.

Lemma 14 $J_{t+1}(w_t) = \lambda\Omega(w_t) + \langle w_t, a_{t+1}\rangle + b_{t+1}$

Proof w_t^b is the optimal value of J on the line joining w_t and w_{t-1}^b while w_t^c is a convex combination of w_t and w_t^b. Moreover by definition of a_{t+1} and b_{t+1} we have $J(w_t^c) = J_{t+1}(w_t^c)$. Therefore,

$$J(w_t^c) = J_{t+1}(w_t^c) = \lambda\Omega(w_t^c) + \langle a_{t+1}, w_t^c\rangle + b_{t+1} \geq J(w_t^b). \tag{35}$$

But since Ω is convex

$$\Omega(\underbrace{(1-\theta)w_t^b + \theta w_t}_{w_t^c}) \leq (1-\theta)\Omega(w_t^b) + \theta\Omega(w_t),$$

which can be rearranged to

$$\theta(\Omega(w_t^b) - \Omega(w_t)) \leq \Omega(w_t^b) - \Omega(w_t^c).$$

Multiplying by λ and adding and subtracting $\theta R_{\text{emp}}(w_t^b)$ and $\theta R_t(w_t)$ respectively to the above equation

$$\underbrace{\lambda\theta\Omega(w_t^b) + \theta R_{\text{emp}}(w_t^b)}_{\theta J(w_t^b)} - \underbrace{\lambda\theta\Omega(w_t) - \theta R_t(w_t)}_{\theta J_t(w_t)} \leq \underbrace{\lambda\Omega(w_t^b) + R_{\text{emp}}(w_t^b)}_{J(w_t^b)} - \lambda\Omega(w_t^c)$$
$$- (1-\theta)R_{\text{emp}}(w_t^b) - \theta R_t(w_t).$$

Plugging in (34) obtains

$$\theta\varepsilon_t \leq J(w_t^b) - \lambda\Omega(w_t^c) - (1-\theta)R_{\text{emp}}(w_t^b) - \theta R_t(w_t). \tag{36}$$

Putting (35) and (36) together

$$\langle a_{t+1}, w_t^c\rangle + b_{t+1} \geq J(w_t^b) - \lambda\Omega(w_t^c) \geq (1-\theta)R_{\text{emp}}(w_t^b) + \theta R_t(w_t) + \theta\varepsilon_t.$$

Since $w_t^c = (1-\theta)w_t^b + \theta w_t$ it follows that

$$(1-\theta)\left\langle a_{t+1}, w_t^b\right\rangle + \theta\left\langle a_{t+1}, w_t\right\rangle + b_{t+1} \geq (1-\theta)R_{\text{emp}}(w_t^b) + \theta R_t(w_t) + \theta\varepsilon_t.$$

Which can be rearranged to

$$(1-\theta)\left(\left\langle a_{t+1}, w_t^b\right\rangle - R_{\text{emp}}(w_t^b)\right) + \theta\left(\langle a_{t+1}, w_t\rangle - R_t(w_t)\right) + b_{t+1} \geq \theta\varepsilon_t.$$

Since $\left\langle w_t^b, a_{t+1}\right\rangle + b_{t+1}$ is the Taylor approximation of the convex function R_{emp} around w_t^c evaluated at w_t^b it follows that $R_{\text{emp}}(w_t^b) \geq \left\langle w_t^b, a_{t+1}\right\rangle + b_{t+1}$. Plugging this into the above equation yields

$$(1-\theta)(-b_{t+1}) + \theta(\langle w_t, a_{t+1}\rangle - R_t(w_t)) + b_{t+1} \geq \theta\varepsilon_t.$$

Dividing by $\theta > 0$ and rearranging yields

$$\langle w_t, a_{t+1}\rangle + b_{t+1} \geq R_t(w_t) + \varepsilon_t.$$

The conclusion of the lemma follows from observing that $R_{t+1}(w_t) = \max(\langle w_t, a_{t+1}\rangle + b_{t+1}, R_t(w_t)) = \langle w_t, a_{t+1}\rangle + b_{t+1}$ and $J_{t+1}(w_t) = \lambda\Omega(w_t) + R_{t+1}(w_t)$. ■

We also need the following two lemmas before we can proceed to the final proof.

Lemma 15 $\varepsilon_t - \varepsilon_{t+1} \geq J_{t+1}(w_{t+1}) - J_t(w_t)$

Proof

$$\begin{aligned}
\varepsilon_t - \varepsilon_{t+1} &= J(w_t^b) - J_t(w_t) - J(w_{t+1}^b) + J_{t+1}(w_{t+1}) \\
&= \underbrace{(J(w_t^b) - J(w_{t+1}^b))}_{\geq 0} + J_{t+1}(w_{t+1}) - J_t(w_t) \quad \text{(by the definition of } w_t^b) \\
&\geq J_{t+1}(w_{t+1}) - J_t(w_t).
\end{aligned}$$

■

Lemma 16 *Let α_t, $\bar{A} := [a_1, \ldots, a_{t+1}]$, and $\bar{b} := [b_1, \ldots, b_{t+1}]$ be as defined in Lemma 11. Then under the assumption of Theorem 4 that* $\max_{u \in \partial_w R_{\mathrm{emp}}(w)} \|u\| \leq G$, *we have*

$$[-\alpha_t, 1]^\top \bar{A}^\top \bar{A} [-\alpha_t, 1] \leq 4G^2.$$

Proof By the dual connection, $\partial_w \lambda \Omega(w_t) = -A\alpha_t$. Also, $\alpha_t \geq 0$, and $\|\alpha_t\|_1 = 1$ as it is the optimal solution of (10) at iteration t. It follows that $\partial_w \lambda \Omega(w_t)$ lies in the convex hull of $a_{t'} \in \partial_w R_{\mathrm{emp}}(w^c_{t'})\ \forall t' \leq t$. Therefore $\|\partial_w \lambda \Omega(w_t)\| \leq G$. Consequently,

$$\begin{aligned}[-\alpha_t, 1]^\top \bar{A}^\top \bar{A} [-\alpha_t, 1] &= \|\partial_w \lambda \Omega(w_t) + a_{t+1}\|^2 \\ &= \|\partial_w \lambda \Omega(w_t)\|^2 + 2\partial_w \lambda \Omega(w_t)^\top a_{t+1} + \|a_{t+1}\|^2 \leq 4G^2,\end{aligned}$$

by Cauchy-Schwarz inequality. ■

Finally, we sketch the proof for Theorem 7.

Proof [Theorem 7] (Sketch) Theorem 4 holds for Algorithm 3 by applying Lemmas 14, 15, and 16 into the first part of the proof. Therefore, for $\epsilon < 4G^2H^*/\lambda$, (33) reduces to $\epsilon_t - \epsilon_{t+1} \geq \lambda\epsilon_t/4G^2H^*$. Applying Lemma 13 yields $\epsilon_t \leq \frac{1}{c\left(t-1+\frac{1}{\epsilon_1 c}\right)}$, with $c = \lambda/8G^2H^*$. Setting $\frac{1}{c\left(t-1+\frac{1}{\epsilon_1 c}\right)} = \epsilon$, assuming that $\epsilon_1 > 0$, and solving for n yields $n \leq \frac{1}{c\epsilon} = \frac{8G^2H^*}{\lambda\epsilon}$. ■

Appendix C. L_1 Regularized BMRM

Following our convention, the L_1 norm regularized BMRM reads

$$\min_{\xi, w} \ \xi + \lambda \|w\|_1 \ \text{subject to } w^\top a_i + b_i \leq \xi,\ i = 1, \ldots, t. \tag{37}$$

An equivalent formulation is

$$\min_{\xi, w} \ \xi \ \text{subject to } w^\top a_i + b_i \leq \xi,\ i = 1, \ldots, t \text{ and } \|w\|_1 \leq \tau, \tag{38}$$

where one can show a monotone correspondence between τ and the λ in (37) by comparison of the KKT conditions for the two problems.

Note that our convergence proof does not apply in this case as the Fenchel dual of $\Omega(w) = \|w\|_1$ fails to satisfy the strong convexity assumption. Nevertheless, we see that (38) can be easily solved by CPM where the solution must lie in the L_1 ball of radius τ. Finally, we note that the L_1 regularized BMRM can be written in a rather standard linear programming (LP) formulation:

$$\begin{aligned}\min_{\xi, u, v} \ & \xi + \lambda \mathbf{1}_d^\top (u + v) \\ \text{s.t. } & a_i^\top u - a_i^\top v + b_i \leq \xi,\ i = 1, \ldots, t \\ & u, v \geq 0,\end{aligned}$$

with the variable of interest $w = u - v$.

References

N. Abe, J. Takeuchi, and M. K. Warmuth. Polynomial Learnability of Stochastic Rules with Respect to the KL-Divergence and Quadratic Distance. *IEICE Transactions on Information and Systems*, 84(3):299–316, 2001.

G. Amdahl. Validity of the single processor approach to achieving large-scale computing capabilities. In *AFIPS Conference Proceedings*, volume 30, pages 483–485, 1967.

M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature Genetics*, 25:25–29, 2000.

G. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data*. MIT Press, Cambridge, Massachusetts, 2007.

O. E. Barndorff-Nielsen. *Information and Exponential Families in Statistical Theory*. John Wiley and Sons, New York, 1978.

J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the International Conference on Machine Learning*, pages 65–72, New York, NY, 2004. ACM Press.

A. Belloni. Introduction to bundle methods. Technical report, Operation Research Center, M.I.T., 2005.

S. Belongie, J. Malik, and J. Puzicha. Matching shapes. In *Eighth IEEE International Conference on Computer Vision*, Vancouver, Canada, July 2001.

K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods Software*, 1:23–34, 1992.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, England, 2004.

J. S. Breese, D. Heckerman, and C. Kardie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

T. Caetano, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. In *Proceedings of the 11th International Conference On Computer Vision*, pages 1–8, Los Alamitos, CA, 2007. IEEE Computer Society.

T. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008. submitted.

L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the Thirteenth ACM conference on Information and knowledge management*, pages 78–87, New York, NY, USA, 2004. ACM Press.

E. Candes and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

C. C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

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

M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, AdaBoost and Bregman distances. In *Proceedings of the 13th Annual Conference on Computational Learning Theory*, pages 158–169. Morgan Kaufmann, San Francisco, 2000.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

R. Cowell, A. Dawid, S. Lauritzen, and D. Spiegelhalter. *Probabilistic Networks and Expert Sytems*. Springer, New York, 1999.

K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, January 2003.

K. Crammer and Y. Singer. Online ranking by projecting. *Neural Computation*, 17(1):145–175, 2005.

N. A. C. Cressie. *Statistics for Spatial Data*. John Wiley and Sons, New York, 1993.

L. Fahrmeir and G. Tutz. *Multivariate Statistical Modelling Based on Generalized Linear Models*. Springer, 1994.

R.-E. Fan, J.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: a library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, August 2008.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, Dec 2001.

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. In A. McCallum and S. Roweis, editors, *Proceedings of the International Conference on Machine Learning*, pages 320–327. Omnipress, 2008.

A. Frangioni. *Dual-Ascent Methods and Multicommodity Flow Problems*. PhD thesis, Dipartimento di Informatica, Universita’ di Pisa, 1997. TD 5/97.

W. Gropp, E. Lusk, and R. Thakur. *Using MPI-2*. MIT Press, 1999.

R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 115–132, Cambridge, MA, 2000. MIT Press.

J. B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms, I and II*, volume 305 and 306. Springer-Verlag, 1993.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In W. Cohen, A. McCallum, and S. Roweis, editors, *icml*, pages 408–415. ACM, 2008.

K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *ACM Special Interest Group in Information Retrieval (SIGIR)*, pages 41–48, New York, 2002. ACM.

T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning*, pages 377–384, San Francisco, California, 2005. Morgan Kaufmann Publishers.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2006.

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.

T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane traning of structural SVMs. *Machine Learning*, 76(1), 2009.

R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987.

M. I. Jordan. *An Introduction to Probabilistic Graphical Models*. MIT Press, 2002. To Appear.

R. M. Karp. An algorithm to solve the $m \times n$ assignment problem in expected time $O(mn \log n)$. *Networks*, 10(2):143–152, 1980.

S. S. Keerthi and D. DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

J. E. Kelly. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, December 1960.

K. C. Kiwiel. An aggregate subgradient method for nonsmooth convex minimization. *Mathematical Programming*, 27:320–341, 1983.

K. C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, 46:105–122, 1990.

R. Koenker. *Quantile Regression*. Cambridge University Press, 2005.

H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

J. D. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic modeling for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, volume 18, pages 282–289, San Francisco, CA, 2001. Morgan Kaufmann.

Q. V. Le and A. J. Smola. Direct optimization of ranking measures. *Journal of Machine Learning Research*, 2007. submitted.

C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Mathematical Programming*, 69:111–147, 1995.

C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650, April 2008.

L. Lukšan and J. Vlček. A bundle-Newton method for nonsmooth unconstrained minimization. *Mathematical Programming*, 83(3):373–391, 1998.

O. L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.

D. McAllester. Generalization bounds and consistency for structured labeling. In *Predicting Structured Data*, Cambridge, Massachusetts, 2007. MIT Press.

T. P. Minka. A comparison of numerical optimizers for logistic regression. Technical report, Microsoft Research, 2007.

K.-R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Artificial Neural Networks ICANN'97*, volume 1327 of *Lecture Notes in Comput. Sci.*, pages 999–1004, Berlin, 1997. Springer-Verlag.

J. Munkres. Algorithms for the assignment and transportation problems. *Journal of SIAM*, 5(1): 32–38, 1957.

A. Nedich and D. P Bertsekas. Convergence rate of incremental subgradient algorithms. In S. Uryasev and P. M. Pardalos, editors, *Stochastic Optimization: Algorithms and Applications*, pages 263–304. Kluwer Academic Publishers, 2000.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

J. B. Orlin and Y. Lee. Quickmatch: a very fast algorithm for the assignment problem. Working Paper 3547-93, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, March 1993.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

N. Ratliff, J. Bagnell, and M. Zinkevich. Maximum margin planning. In *Proceedings of the International Conference on Machine Learning*, July 2006.

N. Ratliff, J. Bagnell, and M. Zinkevich. (online) subgradient methods for structured prediction. In *Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*, March 2007.

G. Rätsch, S. Sonnenburg, J. Srinivasan, H. Witte, K.-R. Müller, R. J. Sommer, and B. Schölkopf. Improving the Caenorhabditis elegans genome annotation using machine learning. *PLoS Computational Biology*, 3(2):e20 doi:10.1371/journal.pcbi.0030020, 2007.

B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

H. Schramm and J. Zowe. A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. *SIAM Journal of Optimization*, 2:121–152, 1992.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL*, pages 213–220, Edmonton, Canada, 2003. Association for Computational Linguistics.

S. Shalev-Schwartz and N. Srebro. SVM optimization: inverse dependence on training set size. In W. Cohen, A. McCallum, and S. Roweis, editors, *Proceedings of the International Conference on Machine Learning*. Omnipress, 2008.

S. Shalev-Shwartz and Y. Singer. Online learning meets optimization in the dual. In H.U. Simon and G. Lugosi, editors, *Computational Learning Theory (COLT)*, LNCS. Springer, 2006. extended version.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: primal estimated sub-gradient solver for SVM. In *Proceedings of the International Conference on Machine Learning*, 2007.

Q. Shi, L. Wang, L. Cheng, and A. J. Smola. Discriminative human action segmentation and recognition using semi-markov model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, 2008.

V. Sindhwani and S. S. Keerthi. Large scale semi-supervised linear SVMs. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 477–484, New York, NY, USA, 2006. ACM Press.

A. J. Smola, S. V. N. Vishwanathan, and Q. V. Le. Bundle methods for machine learning. In D. Koller and Y. Singer, editors, *Advances in Neural Information Processing Systems 20*, Cambridge MA, 2007. MIT Press.

I. Takeuchi, Q. V. Le, T. Sears, and A. J. Smola. Nonparametric quantile estimation. *Journal of Machine Learning Research*, 7, 2006.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32, Cambridge, MA, 2004. MIT Press.

C.-H. Teo, Q. V. Le, A. J. Smola, and S. V. N. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2007.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Section B (Statistical Methodology)*, 58:267–288, 1996.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

V. Vapnik, S. Golowich, and A. J. Smola. Support vector method for function approximation, regression estimation, and signal processing. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 281–287, Cambridge, MA, 1997. MIT Press.

E. Voorhees. Overview of the TREC 2001 question answering track. In *TREC*, 2001.

G. Wahba. Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. Technical Report 984, Department of Statistics, University of Wisconsin, Madison, 1997.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. Technical Report 649, UC Berkeley, Department of Statistics, September 2003.

C. K. I. Williams. Prediction with Gaussian processes: from linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning and Inference in Graphical Models*, pages 599–621. Kluwer Academic, 1998.

J. Yu, S. V. N. Vishwanathan, S. Günter, and N. N. Schraudolph. A quasi-Newton approach to nonsmooth convex optimization. In A. McCallum and S. Roweis, editors, *ICML*, pages 1216–1223. Omnipress, 2008.

T. Zhang. Sequential greedy approximation for certain convex optimization problems. *IEEE Transactions on Information Theory*, 49(3):682–691, 2003.