Journal of Machine Learning Research 11 (2010) 1145-1200 Submitted 11/08; Revised 11/09; Published 3/10

A Quasi-Newton Approach to Nonsmooth Convex Optimization Problems in Machine Learning

Jin Yu JIN.YU@ADELAIDE.EDU.AU
School of Computer Science
The University of Adelaide
Adelaide SA 5005, Australia

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

Simon Günter GUENTER_SIMON@HOTMAIL.COM
DV Bern AG
Nussbaumstrasse 21, CH-3000 Bern 22, Switzerland

Nicol N. Schraudolph JMLR@SCHRAUDOLPH.ORG
adaptive tools AG
Canberra ACT 2602, Australia

Editor: Sathiya Keerthi

Abstract

We extend the well-known BFGS quasi-Newton method and its memory-limited variant LBFGS to the optimization of nonsmooth convex objectives. This is done in a rigorous fashion by generalizing three components of BFGS to subdifferentials: the local quadratic model, the identification of a descent direction, and the Wolfe line search conditions. We prove that under some technical conditions, the resulting subBFGS algorithm is globally convergent in objective function value. We apply its memory-limited variant (subLBFGS) to L_2-regularized risk minimization with the binary hinge loss. To extend our algorithm to the multiclass and multilabel settings, we develop a new, efficient, exact line search algorithm. We prove its worst-case time complexity bounds, and show that our line search can also be used to extend a recently developed bundle method to the multiclass and multilabel settings. We also apply the direction-finding component of our algorithm to L_1-regularized risk minimization with logistic loss. In all these contexts our methods perform comparable to or better than specialized state-of-the-art solvers on a number of publicly available data sets. An open source implementation of our algorithms is freely available.

Keywords: BFGS, variable metric methods, Wolfe conditions, subgradient, risk minimization, hinge loss, multiclass, multilabel, bundle methods, BMRM, OCAS, OWL-QN

1. Introduction

The BFGS quasi-Newton method (Nocedal and Wright, 1999) and its memory-limited LBFGS variant are widely regarded as the workhorses of smooth nonlinear optimization due to their combination of computational efficiency and good asymptotic convergence. Given a smooth objective

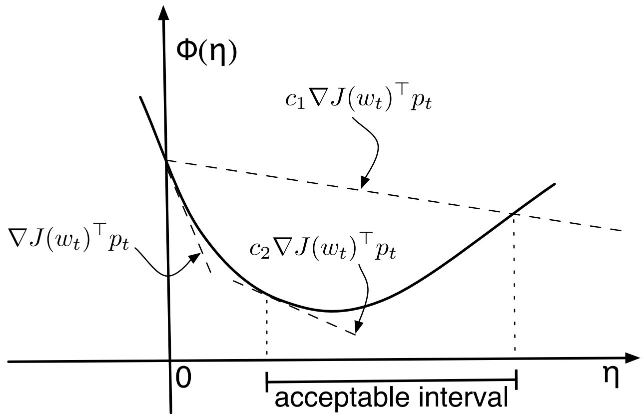


Figure 1: Geometric illustration of the Wolfe conditions (4) and (5).

function $J : \mathbb{R}^d \to \mathbb{R}$ and a current iterate $\boldsymbol{w}_t \in \mathbb{R}^d$, BFGS forms a local quadratic model of J:

$$Q_t(\boldsymbol{p}) \;:=\; J(\boldsymbol{w}_t) + \tfrac{1}{2}\,\boldsymbol{p}^\top \boldsymbol{B}_t^{-1}\boldsymbol{p} + \nabla J(\boldsymbol{w}_t)^\top \boldsymbol{p}\,, \tag{1}$$

where $\boldsymbol{B}_t \succ 0$ is a positive-definite estimate of the inverse Hessian of J, and ∇J denotes the gradient. Minimizing $Q_t(\boldsymbol{p})$ gives the quasi-Newton direction

$$\boldsymbol{p}_t := -\boldsymbol{B}_t \nabla J(\boldsymbol{w}_t), \tag{2}$$

which is used for the parameter update:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta_t \boldsymbol{p}_t. \tag{3}$$

The step size $\eta_t > 0$ is normally determined by a line search obeying the Wolfe (1969) conditions:

$$J(\boldsymbol{w}_{t+1}) \;\le\; J(\boldsymbol{w}_t) + c_1 \eta_t \nabla J(\boldsymbol{w}_t)^\top \boldsymbol{p}_t \qquad \text{(sufficient decrease)} \tag{4}$$

$$\text{and} \quad \nabla J(\boldsymbol{w}_{t+1})^\top \boldsymbol{p}_t \;\ge\; c_2 \nabla J(\boldsymbol{w}_t)^\top \boldsymbol{p}_t \qquad \text{(curvature)} \tag{5}$$

with $0 < c_1 < c_2 < 1$. Figure 1 illustrates these conditions geometrically. The matrix $\boldsymbol{B}_t$ is then modified via the incremental rank-two update

$$\boldsymbol{B}_{t+1} = (\boldsymbol{I} - \rho_t \boldsymbol{s}_t \boldsymbol{y}_t^\top)\boldsymbol{B}_t(\boldsymbol{I} - \rho_t \boldsymbol{y}_t \boldsymbol{s}_t^\top) + \rho_t \boldsymbol{s}_t \boldsymbol{s}_t^\top, \tag{6}$$

where $\boldsymbol{s}_t := \boldsymbol{w}_{t+1} - \boldsymbol{w}_t$ and $\boldsymbol{y}_t := \nabla J(\boldsymbol{w}_{t+1}) - \nabla J(\boldsymbol{w}_t)$ denote the most recent step along the optimization trajectory in parameter and gradient space, respectively, and $\rho_t := (\boldsymbol{y}_t^\top \boldsymbol{s}_t)^{-1}$. The BFGS update (6) enforces the secant equation $\boldsymbol{B}_{t+1}\boldsymbol{y}_t = \boldsymbol{s}_t$. Given a descent direction $\boldsymbol{p}_t$, the Wolfe conditions ensure that $(\forall t)\ \boldsymbol{s}_t^\top \boldsymbol{y}_t > 0$ and hence $\boldsymbol{B}_0 \succ 0 \implies (\forall t)\ \boldsymbol{B}_t \succ 0$.

Limited-memory BFGS (LBFGS, Liu and Nocedal, 1989) is a variant of BFGS designed for high-dimensional optimization problems where the $O(d^2)$ cost of storing and updating $\boldsymbol{B}_t$ would be prohibitive. LBFGS approximates the quasi-Newton direction (2) directly from the last m pairs of $\boldsymbol{s}_t$ and $\boldsymbol{y}_t$ via a matrix-free approach, reducing the cost to $O(md)$ space and time per iteration, with m freely chosen.

There have been some attempts to apply (L)BFGS directly to nonsmooth optimization problems, in the hope that they would perform well on nonsmooth functions that are convex and differentiable almost everywhere. Indeed, it has been noted that in cases where BFGS (resp., LBFGS) does not encounter any nonsmooth point, it often converges to the optimum (Lemarechal, 1982; Lewis and Overton, 2008a). However, Lukšan and Vlček (1999), Haarala (2004), and Lewis and Overton (2008b) also report catastrophic failures of (L)BFGS on nonsmooth functions. Various fixes can be used to avoid this problem, but only in an ad-hoc manner. Therefore, subgradient-based approaches such as subgradient descent (Nedić and Bertsekas, 2000) or bundle methods (Joachims, 2006; Franc and Sonnenburg, 2008; Teo et al., 2010) have gained considerable attention for minimizing nonsmooth objectives.

Although a convex function might not be differentiable everywhere, a subgradient always exists (Hiriart-Urruty and Lemaréchal, 1993). Let $\boldsymbol{w}$ be a point where a convex function J is finite. Then a subgradient is the normal vector of any tangential supporting hyperplane of J at $\boldsymbol{w}$. Formally, $\boldsymbol{g}$ is called a subgradient of J at $\boldsymbol{w}$ if and only if (Hiriart-Urruty and Lemaréchal, 1993, Definition VI.1.2.1)

$$(\forall \boldsymbol{w}')\ \ J(\boldsymbol{w}')\ \geq\ J(\boldsymbol{w}) + (\boldsymbol{w}' - \boldsymbol{w})^\top \boldsymbol{g}. \tag{7}$$

The set of all subgradients at a point is called the subdifferential, and is denoted $\partial J(\boldsymbol{w})$. If this set is not empty then J is said to be *subdifferentiable at* $\boldsymbol{w}$. If it contains exactly one element, that is, $\partial J(\boldsymbol{w}) = \{\nabla J(\boldsymbol{w})\}$, then J is *differentiable* at $\boldsymbol{w}$. Figure 2 provides the geometric interpretation of (7).

The aim of this paper is to develop principled and robust quasi-Newton methods that are amenable to subgradients. This results in subBFGS and its memory-limited variant subLBFGS, two new subgradient quasi-Newton methods that are applicable to nonsmooth convex optimization problems. In particular, we apply our algorithms to a variety of machine learning problems, exploiting knowledge about the subdifferential of the binary hinge loss and its generalizations to the multiclass and multilabel settings.

In the next section we motivate our work by illustrating the difficulties of LBFGS on nonsmooth functions, and the advantage of incorporating BFGS' curvature estimate into the parameter update. In Section 3 we develop our optimization algorithms generically, before discussing their application to L_2-regularized risk minimization with the hinge loss in Section 4. We describe a new efficient algorithm to identify the nonsmooth points of a one-dimensional pointwise maximum of linear functions in Section 5, then use it to develop an exact line search that extends our optimization algorithms to the multiclass and multilabel settings (Section 6). Section 7 compares and contrasts our work with other recent efforts in this area. We report our experimental results on a number of public data sets in Section 8, and conclude with a discussion and outlook in Section 9.

2. Motivation

The application of standard (L)BFGS to nonsmooth optimization is problematic since the quasi-Newton direction generated at a nonsmooth point is not necessarily a descent direction. Nevertheless, BFGS' inverse Hessian estimate can provide an effective model of the overall shape of a nonsmooth objective; incorporating it into the parameter update can therefore be beneficial. We

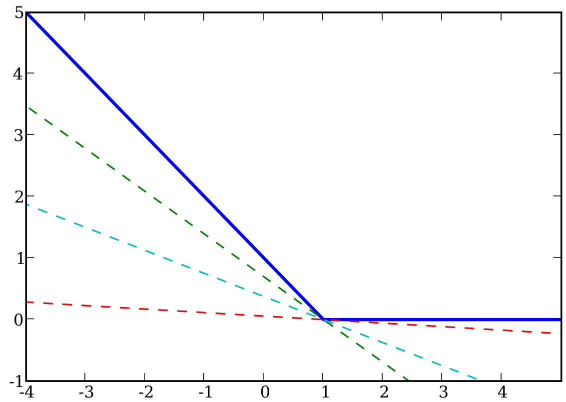


Figure 2: Geometric interpretation of subgradients. The dashed lines are tangential to the hinge function (solid blue line); the slopes of these lines are subgradients.

discuss these two aspects of (L)BFGS to motivate our work on developing new quasi-Newton methods that are amenable to subgradients while preserving the fast convergence properties of standard (L)BFGS.

2.1 Problems of (L)BFGS on Nonsmooth Objectives

Smoothness of the objective function is essential for classical (L)BFGS because both the local quadratic model (1) and the Wolfe conditions (4, 5) require the existence of the gradient ∇J at every point. As pointed out by Hiriart-Urruty and Lemaréchal (1993, Remark VIII.2.1.3), even though nonsmooth convex functions are differentiable everywhere except on a set of Lebesgue measure zero, it is unwise to just use a smooth optimizer on a nonsmooth convex problem under the assumption that "it should work almost surely." Below we illustrate this on both a toy example and real-world machine learning problems.

2.1.1 A Toy Example

The following simple example demonstrates the problems faced by BFGS when working with a nonsmooth objective function, and how our subgradient BFGS (subBFGS) method (to be introduced in Section 3) with exact line search overcomes these problems. Consider the task of minimizing

$$f(x, y) = 10\,|x| + |y| \tag{8}$$

with respect to x and y. Clearly, $f(x, y)$ is convex but nonsmooth, with the minimum located at $(0, 0)$ (Figure 3, left). It is subdifferentiable whenever x or y is zero:

$$\partial_x f(0, \cdot) = [-10, 10] \text{ and } \partial_y f(\cdot, 0) = [-1, 1].$$

We call such lines of subdifferentiability in parameter space *hinges*.

We can minimize (8) with the standard BFGS algorithm, employing a backtracking line search (Nocedal and Wright, 1999, Procedure 3.1) that starts with a step size that obeys the curvature

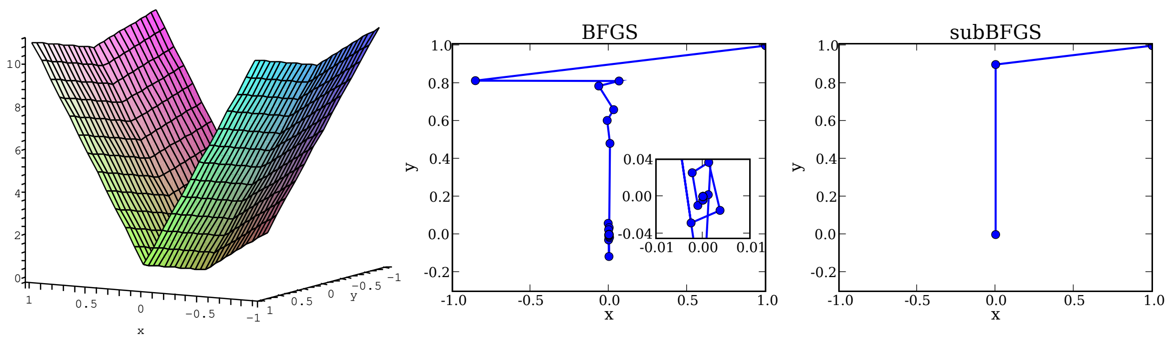


Figure 3: Left: the nonsmooth convex function (8); optimization trajectory of BFGS with inexact line search (center) and subBFGS (right) on this function.

condition (5), then exponentially decays it until both Wolfe conditions (4, 5) are satisfied.[1] The curvature condition forces BFGS to jump across at least one hinge, thus ensuring that the gradient displacement vector $\boldsymbol{y}_t$ in (6) is non-zero; this prevents BFGS from diverging. Moreover, with such an *inexact* line search BFGS will generally not step on any hinges directly, thus avoiding (in an ad-hoc manner) the problem of non-differentiability. Although this algorithm quickly decreases the objective from the starting point $(1, 1)$, it is then slowed down by heavy oscillations around the optimum (Figure 3, center), caused by the utter mismatch between BFGS' quadratic model and the actual function.

A generally sensible strategy is to use an exact line search that finds the optimum along a given descent direction (cf. Section 4.2.1). However, this line optimum will often lie on a hinge (as it does in our toy example), where the function is not differentiable. If an arbitrary subgradient is supplied instead, the BFGS update (6) can produce a search direction which is not a descent direction, causing the next line search to fail. In our toy example, standard BFGS with exact line search consistently fails after the first step, which takes it to the hinge at $x = 0$.

Unlike standard BFGS, our subBFGS method can handle hinges and thus reap the benefits of an exact line search. As Figure 3 (right) shows, once the first iteration of subBFGS lands it on the hinge at $x = 0$, its direction-finding routine (Algorithm 2) finds a descent direction for the next step. In fact, on this simple example Algorithm 2 yields a vector with zero x component, which takes subBFGS straight to the optimum at the second step.[2]

2.1.2 TYPICAL NONSMOOTH OPTIMIZATION PROBLEMS IN MACHINE LEARNING

The problems faced by smooth quasi-Newton methods on nonsmooth objectives are not only encountered in cleverly constructed toy examples, but also in real-world applications. To show this, we apply LBFGS to L_2-regularized risk minimization problems (30) with binary hinge loss (31), a typical nonsmooth optimization problem encountered in machine learning. For this particular objective function, an exact line search is cheap and easy to compute (see Section 4.2.1 for details). Figure 4 (left & center) shows the behavior of LBFGS with this exact line search (LBFGS-LS)

1. We set $c_1 = 10^{-3}$ in (4) and $c_2 = 0.8$ in (5), and used a decay factor of 0.9.
2. This is achieved for any choice of initial subgradient $\boldsymbol{g}^{(1)}$ (Line 3 of Algorithm 2).

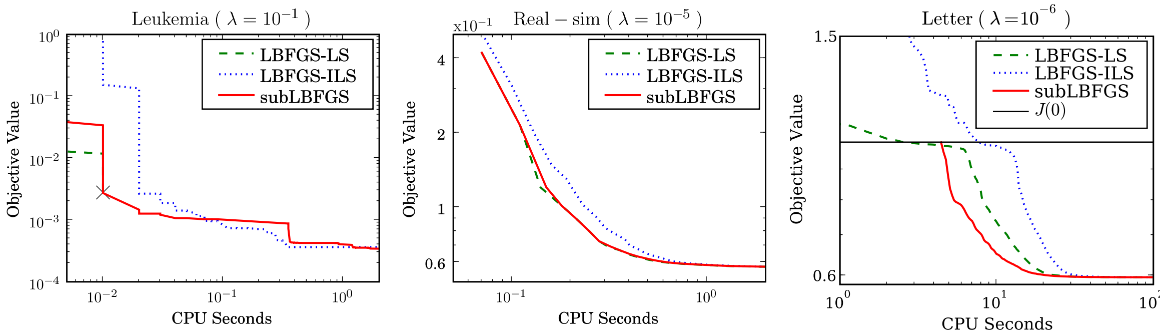


Figure 4: Performance of subLBFGS (solid) and standard LBFGS with exact (dashed) and inexact (dotted) line search methods on sample L_2-regularized risk minimization problems with the binary (left and center) and multiclass hinge losses (right). LBFGS with exact line search (dashed) fails after 3 iterations (marked as $\times$) on the Leukemia data set (left).

on two data sets, namely Leukemia and Real-sim.[3] It can be seen that LBFGS-LS converges on Real-sim but diverges on the Leukemia data set. This is because using an exact line search on a nonsmooth objective function increases the chance of landing on nonsmooth points, a situation that standard BFGS (resp., LBFGS) is not designed to deal with. To prevent (L)BFGS' sudden breakdown, a scheme that actively avoids nonsmooth points must be used. One such possibility is to use an inexact line search that obeys the Wolfe conditions. Here we used an efficient inexact line search that uses a caching scheme specifically designed for L_2-regularized hinge loss (cf. end of Section 4.2). This implementation of LBFGS (LBFGS-ILS) converges on both data sets shown here but may fail on others. It is also slower, due to the inexactness of its line search.

For the multiclass hinge loss (42) we encounter another problem: if we follow the usual practice of initializing $\boldsymbol{w} = \mathbf{0}$, which happens to be a non-differentiable point, then LBFGS stalls. One way to get around this is to force LBFGS to take a unit step along its search direction to escape this nonsmooth point. However, as can be seen on the Letter data set[3] in Figure 4 (right), such an ad-hoc fix increases the value of the objective above $J(\mathbf{0})$ (solid horizontal line), and it takes several CPU seconds for the optimizers to recover from this. In all cases shown in Figure 4, our subgradient LBFGS (subLBFGS) method (as will be introduced later) performs comparable to or better than the best implementation of LBFGS.

2.2 Advantage of Incorporating BFGS' Curvature Estimate

In machine learning one often encounters L_2-regularized risk minimization problems (30) with various hinge losses (31, 42, 55). Since the Hessian of those objective functions at differentiable points equals $\lambda \boldsymbol{I}$ (where λ is the regularization constant), one might be tempted to argue that for such problems, BFGS' approximation $\boldsymbol{B}_t$ to the inverse Hessian should be simply set to $\lambda^{-1}\boldsymbol{I}$. This would reduce the quasi-Newton direction $\boldsymbol{p}_t = -\boldsymbol{B}_t \boldsymbol{g}_t,\ \boldsymbol{g}_t \in \partial J(\boldsymbol{w}_t)$ to simply a scaled subgradient direction.

To check if doing so is beneficial, we compared the performance of our subLBFGS method with two implementations of subgradient descent: a vanilla gradient descent method (denoted GD) that

3. Descriptions of these data sets can be found in Section 8.

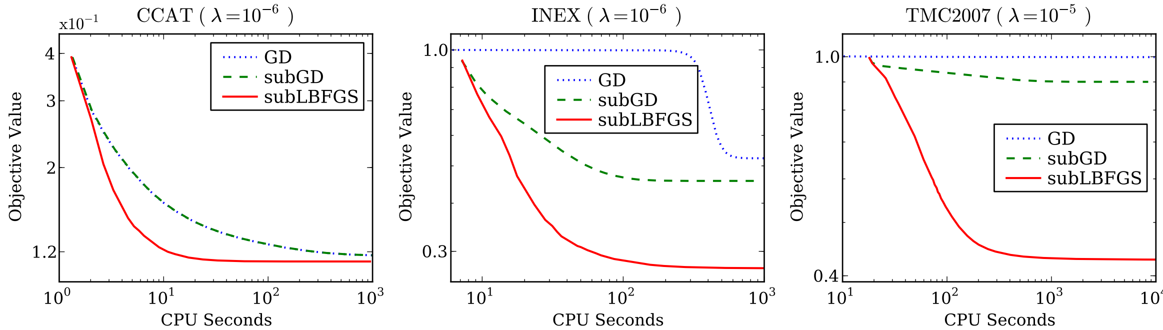


Figure 5: Performance of subLBFGS, GD, and subGD on sample L_2-regularized risk minimization problems with binary (left), multiclass (center), and multilabel (right) hinge losses.

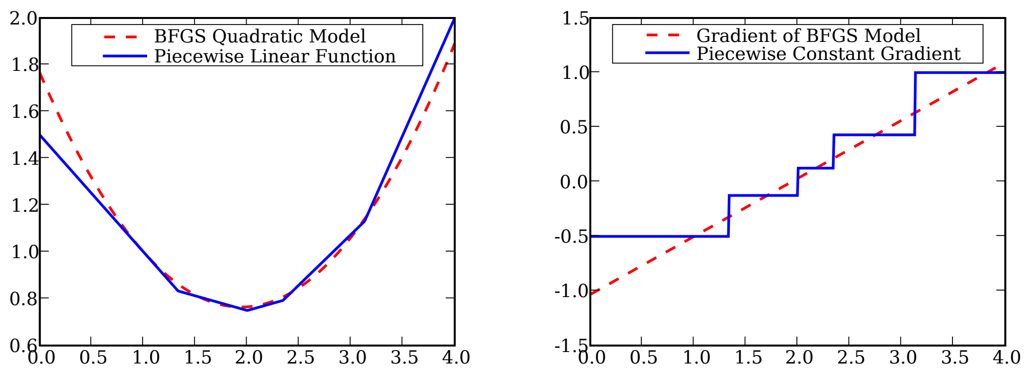


Figure 6: BFGS' quadratic approximation to a piecewise linear function (left), and its estimate of the gradient of this function (right).

uses a random subgradient for its parameter update, and an improved subgradient descent method (denoted subGD) whose parameter is updated in the direction produced by our direction-finding routine (Algorithm 2) with $\boldsymbol{B}_t = \boldsymbol{I}$. All algorithms used exact line search, except that GD took a unit step for the first update in order to avoid the nonsmooth point $\boldsymbol{w}_0 = \mathbf{0}$ (cf. the discussion in Section 2.1). As can be seen in Figure 5, on all sample L_2-regularized hinge loss minimization problems, subLBFGS (solid) converges significantly faster than GD (dotted) and subGD (dashed). This indicates that BFGS' $\boldsymbol{B}_t$ matrix is able to model the objective function, including its hinges, better than simply setting $\boldsymbol{B}_t$ to a scaled identity matrix.

We believe that BFGS' curvature update (6) plays an important role in the performance of subLBFGS seen in Figure 5. Recall that (6) satisfies the secant condition $\boldsymbol{B}_{t+1}\boldsymbol{y}_t = \boldsymbol{s}_t$, where $\boldsymbol{s}_t$ and $\boldsymbol{y}_t$ are displacement vectors in parameter and gradient space, respectively. The secant condition in fact implements a *finite differencing* scheme: for a one-dimensional objective function $J : \mathbb{R} \to \mathbb{R}$, we have

$$\boldsymbol{B}_{t+1} = \frac{(w+p)-w}{\nabla J(w+p) - \nabla J(w)}. \tag{9}$$

Although the original motivation behind the secant condition was to approximate the inverse Hessian, the finite differencing scheme (9) allows BFGS to model the global curvature (i.e., overall shape) of the objective function from first-order information. For instance, Figure 6 (left) shows that the BFGS quadratic model[4] (1) fits a piecewise linear function quite well despite the fact that the actual Hessian in this case is zero almost everywhere, and infinite (in the limit) at nonsmooth points. Figure 6 (right) reveals that BFGS captures the global trend of the gradient rather than its infinitesimal variation, that is, the Hessian. This is beneficial for nonsmooth problems, where Hessian does not fully represent the overall curvature of the objective function.

3. Subgradient BFGS Method

We modify the standard BFGS algorithm to derive our new algorithm (subBFGS, Algorithm 1) for nonsmooth convex optimization, and its memory-limited variant (subLBFGS). Our modifications can be grouped into three areas, which we elaborate on in turn: generalizing the local quadratic model, finding a descent direction, and finding a step size that obeys a subgradient reformulation of the Wolfe conditions. We then show that our algorithm's estimate of the inverse Hessian has a bounded spectrum, which allows us to prove its convergence.

Algorithm 1 Subgradient BFGS (subBFGS)

Initialize: $t := 0, \boldsymbol{w}_0 = \boldsymbol{0}, \boldsymbol{B}_0 = \boldsymbol{I}$
Set: direction-finding tolerance $\epsilon \geq 0$, iteration limit $k_{\text{max}} > 0$,
  lower bound $h > 0$ on $\frac{\boldsymbol{s}_t^\top \boldsymbol{y}_t}{\boldsymbol{y}_t^\top \boldsymbol{y}_t}$ (cf. discussion in Section 3.4)
Compute subgradient $\boldsymbol{g}_0 \in \partial J(\boldsymbol{w}_0)$
while not converged **do**
  $\boldsymbol{p}_t = \texttt{descentDirection}(\boldsymbol{g}_t, \epsilon, k_{\text{max}})$ (Algorithm 2)
  if $\boldsymbol{p}_t =$ failure **then**
    Return $\boldsymbol{w}_t$
  end if
  Find η_t that obeys (23) and (24) (e.g., Algorithm 3 or 5)
  $\boldsymbol{s}_t = \eta_t \boldsymbol{p}_t$
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{s}_t$
  Choose subgradient $\boldsymbol{g}_{t+1} \in \partial J(\boldsymbol{w}_{t+1}) : \boldsymbol{s}_t^\top(\boldsymbol{g}_{t+1} - \boldsymbol{g}_t) > 0$
  $\boldsymbol{y}_t := \boldsymbol{g}_{t+1} - \boldsymbol{g}_t$
  $\boldsymbol{s}_t := \boldsymbol{s}_t + \max\left(0,\ h - \frac{\boldsymbol{s}_t^\top \boldsymbol{y}_t}{\boldsymbol{y}_t^\top \boldsymbol{y}_t}\right) \boldsymbol{y}_t$ (ensure $\frac{\boldsymbol{s}_t^\top \boldsymbol{y}_t}{\boldsymbol{y}_t^\top \boldsymbol{y}_t} \geq h$)
  Update $\boldsymbol{B}_{t+1}$ via (6)
  $t := t+1$
end while

4. For ease of exposition, the model was constructed at a differentiable point.

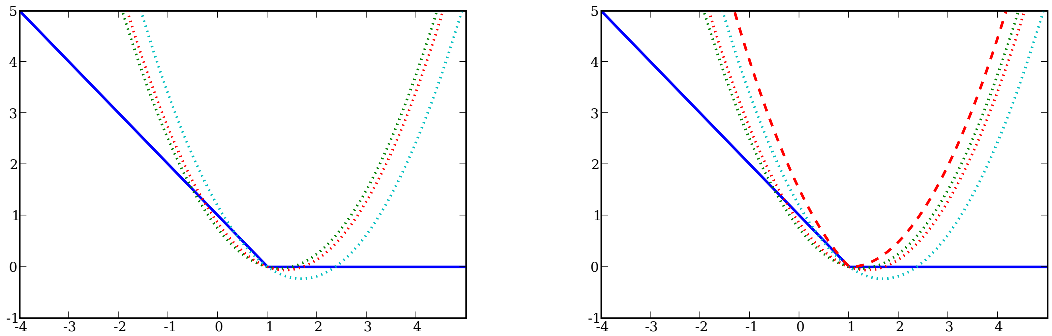


Figure 7: Left: selecting arbitrary subgradients yields many possible quadratic models (dotted lines) for the objective (solid blue line) at a subdifferentiable point. The models were built by keeping $\boldsymbol{B}_t$ fixed, but selecting random subgradients. Right: the tightest pseudo-quadratic fit (10) (bold red dashes); note that it is not a quadratic.

3.1 Generalizing the Local Quadratic Model

Recall that BFGS assumes that the objective function J is differentiable everywhere so that at the current iterate $\boldsymbol{w}_t$ it can construct a local quadratic model (1) of $J(\boldsymbol{w}_t)$. For a nonsmooth objective function, such a model becomes ambiguous at non-differentiable points (Figure 7, left). To resolve the ambiguity, we could simply replace the gradient $\nabla J(\boldsymbol{w}_t)$ in (1) with an arbitrary subgradient $\boldsymbol{g}_t \in \partial J(\boldsymbol{w}_t)$. However, as will be discussed later, the resulting quasi-Newton direction $\boldsymbol{p}_t := -\boldsymbol{B}_t \boldsymbol{g}_t$ is not necessarily a descent direction. To address this fundamental modeling problem, we first generalize the local quadratic model (1) as follows:

$$\begin{aligned} Q_t(\boldsymbol{p}) &:= J(\boldsymbol{w}_t) + M_t(\boldsymbol{p}), \quad \text{where} \\ M_t(\boldsymbol{p}) &:= \tfrac{1}{2}\boldsymbol{p}^\top \boldsymbol{B}_t^{-1}\boldsymbol{p} + \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}. \end{aligned} \tag{10}$$

Note that where J is differentiable, (10) reduces to the familiar BFGS quadratic model (1). At non-differentiable points, however, the model is no longer quadratic, as the supremum may be attained at different elements of $\partial J(\boldsymbol{w}_t)$ for different directions $\boldsymbol{p}$. Instead it can be viewed as the tightest pseudo-quadratic fit to J at $\boldsymbol{w}_t$ (Figure 7, right). Although the local model (10) of subBFGS is nonsmooth, it only incorporates non-differential points present at the current location; all others are smoothly approximated by the quasi-Newton mechanism.

Having constructed the model (10), we can minimize $Q_t(\boldsymbol{p})$, or equivalently $M_t(\boldsymbol{p})$:

$$\min_{\boldsymbol{p}\in\mathbb{R}^d} \left(\tfrac{1}{2}\boldsymbol{p}^\top \boldsymbol{B}_t^{-1}\boldsymbol{p} + \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p} \right) \tag{11}$$

to obtain a search direction. We now show that solving (11) is closely related to the problem of finding a *normalized steepest descent* direction. A normalized steepest descent direction is defined as the solution to the following problem (Hiriart-Urruty and Lemaréchal, 1993, Chapter VIII):

$$\min_{\boldsymbol{p}\in\mathbb{R}^d} \; J'(\boldsymbol{w}_t,\, \boldsymbol{p}) \;\text{ s.t. }\; \|\boldsymbol{p}\| \leq 1, \tag{12}$$

where

$$J'(\boldsymbol{w}_t,\, \boldsymbol{p}) := \lim_{\eta\downarrow 0} \frac{J(\boldsymbol{w}_t + \eta\boldsymbol{p}) - J(\boldsymbol{w}_t)}{\eta}$$

is the directional derivative of J at $\boldsymbol{w}_t$ in direction $\boldsymbol{p}$, and $\|\cdot\|$ is a norm defined on $\mathbb{R}^d$. In other words, the normalized steepest descent direction is the direction of bounded norm along which the maximum rate of decrease in the objective function value is achieved. Using the property: $J'(\boldsymbol{w}_t,\, \boldsymbol{p}) = \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top\boldsymbol{p}$ (Bertsekas, 1999, Proposition B.24.b), we can rewrite (12) as:

$$\min_{\boldsymbol{p}\in\mathbb{R}^d} \; \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top\boldsymbol{p} \quad \text{s.t. } \|\boldsymbol{p}\| \leq 1. \tag{13}$$

If the matrix $\boldsymbol{B}_t \succ 0$ as in (11) is used to define the norm $\|\cdot\|$ as

$$\|\boldsymbol{p}\|^2 := \boldsymbol{p}^\top\boldsymbol{B}_t^{-1}\boldsymbol{p}, \tag{14}$$

then the solution to (13) points to the same direction as that obtained by minimizing our pseudo-quadratic model (11). To see this, we write the Lagrangian of the constrained minimization problem (13):

$$\begin{aligned} L(\boldsymbol{p}, \alpha) \; &:= \alpha\, \boldsymbol{p}^\top\boldsymbol{B}_t^{-1}\boldsymbol{p} \; - \alpha \; + \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top\boldsymbol{p} \\ &= \tfrac{1}{2}\, \boldsymbol{p}^\top(2\alpha\boldsymbol{B}_t^{-1})\boldsymbol{p} \; - \alpha \; + \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top\boldsymbol{p}, \end{aligned} \tag{15}$$

where $\alpha > 0$ is a Lagrangian multiplier. It is easy to see from (15) that minimizing the Lagrangian function L with respect to $\boldsymbol{p}$ is equivalent to solving (11) with $\boldsymbol{B}_t^{-1}$ scaled by a scalar 2α, implying that the steepest descent direction obtained by solving (13) with the weighted norm (14) only differs in length from the search direction obtained by solving (11). Therefore, our search direction is essentially an unnomalized steepest descent direction with respect to the weighted norm (14).

Ideally, we would like to solve (11) to obtain the best search direction. This is generally intractable due to the presence a supremum over the entire subdifferential set $\partial J(\boldsymbol{w}_t)$. In many machine learning problems, however, $\partial J(\boldsymbol{w}_t)$ has some special structure that simplifies the calculation of that supremum. In particular, the subdifferential of all the problems considered in this paper is a convex and compact polyhedron characterised as the convex hull of its extreme points. This dramatically reduces the cost of calculating $\sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top\boldsymbol{p}$ since the supremum can only be attained at an extreme point of the polyhedral set $\partial J(\boldsymbol{w}_t)$ (Bertsekas, 1999, Proposition B.21c). In what follows, we develop an iterative procedure that is guaranteed to find a quasi-Newton descent direction, assuming an oracle that supplies $\arg\sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top\boldsymbol{p}$ for a given direction $\boldsymbol{p} \in \mathbb{R}^d$. Efficient oracles for this purpose can be derived for many machine learning settings; we provides such oracles for L_2-regularized risk minimization with the binary hinge loss (Section 4.1), multiclass and multilabel hinge losses (Section 6), and L_1-regularized logistic loss (Section 8.4).

Algorithm 2 $\boldsymbol{p}_t = \texttt{descentDirection}(\boldsymbol{g}^{(1)}, \epsilon, k_{\max})$

input (sub)gradient $\boldsymbol{g}^{(1)} \in \partial J(\boldsymbol{w}_t)$, tolerance $\epsilon \geq 0$, iteration limit $k_{\max} > 0$,
  and an oracle to calculate $\arg\sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w})} \boldsymbol{g}^\top \boldsymbol{p}$ for any given $\boldsymbol{w}$ and $\boldsymbol{p}$
output descent direction $\boldsymbol{p}_t$
Initialize: $i = 1,\ \bar{\boldsymbol{g}}^{(1)} = \boldsymbol{g}^{(1)},\ \boldsymbol{p}^{(1)} = -\boldsymbol{B}_t \boldsymbol{g}^{(1)}$
$\boldsymbol{g}^{(2)} = \arg\sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}^{(1)}$
$\epsilon^{(1)} := \boldsymbol{p}^{(1)\top} \boldsymbol{g}^{(2)} - \boldsymbol{p}^{(1)\top} \bar{\boldsymbol{g}}^{(1)}$
while $(\boldsymbol{g}^{(i+1)\top} \boldsymbol{p}^{(i)} > 0$ or $\epsilon^{(i)} > \epsilon)$ and $\epsilon^{(i)} > 0$ and $i < k_{\max}$ **do**
  $\mu^* := \min \left[1, \frac{(\bar{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i+1)})^\top \boldsymbol{B}_t \bar{\boldsymbol{g}}^{(i)}}{(\bar{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i+1)})^\top \boldsymbol{B}_t (\bar{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i+1)})} \right]$; see (97)
  $\bar{\boldsymbol{g}}^{(i+1)} = (1 - \mu^*) \bar{\boldsymbol{g}}^{(i)} + \mu^* \boldsymbol{g}^{(i+1)}$
  $\boldsymbol{p}^{(i+1)} = (1 - \mu^*) \boldsymbol{p}^{(i)} - \mu^* \boldsymbol{B}_t \boldsymbol{g}^{(i+1)}$; see (76)
  $\boldsymbol{g}^{(i+2)} = \arg\sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}^{(i+1)}$
  $\epsilon^{(i+1)} := \min_{j \leq (i+1)} \left[\boldsymbol{p}^{(j)\top} \boldsymbol{g}^{(j+1)} - \frac{1}{2} (\boldsymbol{p}^{(j)\top} \bar{\boldsymbol{g}}^{(j)} + \boldsymbol{p}^{(i+1)\top} \bar{\boldsymbol{g}}^{(i+1)}) \right]$
  $i := i + 1$
end while
$\boldsymbol{p}_t = \operatorname{argmin}_{j \leq i} M_t(\boldsymbol{p}^{(j)})$
if $\sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}_t \geq 0$ **then**
  return failure;
else
  return $\boldsymbol{p}_t$.
end if

3.2 Finding a Descent Direction

A direction $\boldsymbol{p}_t$ is a descent direction if and only if $\boldsymbol{g}^\top \boldsymbol{p}_t < 0 \ \ \forall \boldsymbol{g} \in \partial J(\boldsymbol{w}_t)$ (Hiriart-Urruty and Lemaréchal, 1993, Theorem VIII.1.1.2), or equivalently

$$\sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}_t \ < \ 0. \tag{16}$$

For a smooth convex function, the quasi-Newton direction (2) is always a descent direction because

$$\nabla J(\boldsymbol{w}_t)^\top \boldsymbol{p}_t \ = \ -\nabla J(\boldsymbol{w}_t)^\top \boldsymbol{B}_t \nabla J(\boldsymbol{w}_t) \ < \ 0$$

holds due to the positivity of $\boldsymbol{B}_t$.

For nonsmooth functions, however, the quasi-Newton direction $\boldsymbol{p}_t := -\boldsymbol{B}_t \boldsymbol{g}_t$ for a given $\boldsymbol{g}_t \in \partial J(\boldsymbol{w}_t)$ may not fulfill the descent condition (16), making it impossible to find a step size $\eta > 0$ that obeys the Wolfe conditions (4, 5), thus causing a failure of the line search. We now present an iterative approach to finding a quasi-Newton *descent* direction.

Our goal is to minimize the pseudo-quadratic model (10), or equivalently minimize $M_t(\boldsymbol{p})$. Inspired by bundle methods (Teo et al., 2010), we achieve this by minimizing convex lower bounds of $M_t(\boldsymbol{p})$ that are designed to progressively approach $M_t(\boldsymbol{p})$ over iterations. At iteration i we build the following convex lower bound on $M_t(\boldsymbol{p})$:

$$M_t^{(i)}(\boldsymbol{p}) \ := \ \tfrac{1}{2} \boldsymbol{p}^\top \boldsymbol{B}_t^{-1} \boldsymbol{p} + \sup_{j \leq i} \boldsymbol{g}^{(j)\top} \boldsymbol{p}, \tag{17}$$

where $i, j \in \mathbb{N}$ and $\boldsymbol{g}^{(j)} \in \partial J(\boldsymbol{w}_t)\ \forall j \leq i$. Given a $\boldsymbol{p}^{(i)} \in \mathbb{R}^d$ the lower bound (17) is successively tightened by computing

$$\boldsymbol{g}^{(i+1)} := \operatorname*{arg\,sup}_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}^{(i)}, \tag{18}$$

such that $M_t^{(i)}(\boldsymbol{p}) \leq M_t^{(i+1)}(\boldsymbol{p}) \leq M_t(\boldsymbol{p})\ \forall \boldsymbol{p} \in \mathbb{R}^d$. Here we set $\boldsymbol{g}^{(1)} \in \partial J(\boldsymbol{w}_t)$ arbitrarily, and assume that (18) is provided by an oracle (e.g., as described in Section 4.1). To solve $\min_{\boldsymbol{p} \in \mathbb{R}^d} M_t^{(i)}(\boldsymbol{p})$, we rewrite it as a constrained optimization problem:

$$\min_{\boldsymbol{p}, \xi} \left(\tfrac{1}{2} \boldsymbol{p}^\top \boldsymbol{B}_t^{-1} \boldsymbol{p} + \xi \right) \quad \text{s.t.}\ \boldsymbol{g}^{(j)\top} \boldsymbol{p} \leq \xi\ \ \forall j \leq i. \tag{19}$$

This problem can be solved exactly via quadratic programming, but doing so may incur substantial computational expense. Instead we adopt an alternative approach (Algorithm 2) which does not solve (19) to optimality. The key idea is to write the proposed descent direction at iteration $i+1$ as a convex combination of $\boldsymbol{p}^{(i)}$ and $-\boldsymbol{B}_t \boldsymbol{g}^{(i+1)}$ (Line 9 of Algorithm 2); and as will be shown in Appendix B, the returned search direction takes the form

$$\boldsymbol{p}_t = -\boldsymbol{B}_t \bar{\boldsymbol{g}}_t,$$

where $\bar{\boldsymbol{g}}_t$ is a subgradient in $\partial J(\boldsymbol{w}_t)$ that allows $\boldsymbol{p}_t$ to satisfy the descent condition (16). The optimal convex combination coefficient μ^* can be computed exactly (Line 7 of Algorithm 2) using an argument based on maximizing the dual objective of $M_t(\boldsymbol{p})$; see Appendix A for details.

The weak duality theorem (Hiriart-Urruty and Lemaréchal, 1993, Theorem XII.2.1.5) states that the optimal primal value is no less than any dual value, that is, if $D_t(\boldsymbol{\alpha})$ is the dual of $M_t(\boldsymbol{p})$, then $\min_{\boldsymbol{p} \in \mathbb{R}^d} M_t(\boldsymbol{p}) \geq D_t(\boldsymbol{\alpha})$ holds for all feasible dual solutions $\boldsymbol{\alpha}$. Therefore, by iteratively increasing the value of the dual objective we close the gap to optimality in the primal. Based on this argument, we use the following upper bound on the duality gap as our measure of progress:

$$\epsilon^{(i)} := \min_{j \leq i} \left[\boldsymbol{p}^{(j)\top} \boldsymbol{g}^{(j+1)} - \tfrac{1}{2} (\boldsymbol{p}^{(j)\top} \bar{\boldsymbol{g}}^{(j)} + \boldsymbol{p}^{(i)\top} \bar{\boldsymbol{g}}^{(i)}) \right] \geq \min_{\boldsymbol{p} \in \mathbb{R}^d} M_t(\boldsymbol{p}) - D_t(\boldsymbol{\alpha}^*), \tag{20}$$

where $\bar{\boldsymbol{g}}^{(i)}$ is an aggregated subgradient (Line 8 of Algorithm 2) which lies in the convex hull of $\boldsymbol{g}^{(j)} \in \partial J(\boldsymbol{w}_t)\ \forall j \leq i$, and $\boldsymbol{\alpha}^*$ is the optimal dual solution; Equations 77–79 in Appendix A provide intermediate steps that lead to the inequality in (20). Theorem 7 (Appendix B) shows that $\epsilon^{(i)}$ is monotonically decreasing, leading us to a practical stopping criterion (Line 6 of Algorithm 2) for our direction-finding procedure.

A detailed derivation of Algorithm 2 is given in Appendix A, where we also prove that at a non-optimal iterate a direction-finding tolerance $\epsilon \geq 0$ exists such that the search direction produced by Algorithm 2 is a descent direction; in Appendix B we prove that Algorithm 2 converges to a solution with precision ϵ in $O(1/\epsilon)$ iterations. Our proofs are based on the assumption that the spectrum (eigenvalues) of BFGS' approximation $\boldsymbol{B}_t$ to the inverse Hessian is bounded from above and below. This is a reasonable assumption if simple safeguards such as those described in Section 3.4 are employed in the practical implementation.

3.3 Subgradient Line Search

Given the current iterate $\boldsymbol{w}_t$ and a search direction $\boldsymbol{p}_t$, the task of a line search is to find a step size $\eta > 0$ which reduces the objective function value along the line $\boldsymbol{w}_t + \eta \boldsymbol{p}_t$:

$$\text{minimize } \Phi(\eta) := J(\boldsymbol{w}_t + \eta \boldsymbol{p}_t). \tag{21}$$

Using the chain rule, we can write

$$\partial \, \Phi(\eta) := \{\boldsymbol{g}^\top \boldsymbol{p}_t : \boldsymbol{g} \in \partial J(\boldsymbol{w}_t + \eta \boldsymbol{p}_t)\}. \tag{22}$$

Exact line search finds the optimal step size η^* by minimizing $\Phi(\eta)$, such that $0 \in \partial \Phi(\eta^*)$; inexact line searches solve (21) approximately while enforcing conditions designed to ensure convergence. The Wolfe conditions (4) and (5), for instance, achieve this by guaranteeing a sufficient decrease in the value of the objective and excluding pathologically small step sizes, respectively (Wolfe, 1969; Nocedal and Wright, 1999). The original Wolfe conditions, however, require the objective function to be smooth; to extend them to nonsmooth convex problems, we propose the following subgradient reformulation:

$$J(\boldsymbol{w}_{t+1}) \leq J(\boldsymbol{w}_t) + c_1 \eta_t \sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}_t \qquad \text{(sufficient decrease)} \tag{23}$$

$$\text{and} \qquad \sup_{\boldsymbol{g}' \in \partial J(\boldsymbol{w}_{t+1})} \boldsymbol{g}'^\top \boldsymbol{p}_t \geq c_2 \sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}_t, \qquad \text{(curvature)} \tag{24}$$

where $0 < c_1 < c_2 < 1$. Figure 8 illustrates how these conditions enforce acceptance of non-trivial step sizes that decrease the objective function value. In Appendix C we formally show that for any given descent direction we can always find a positive step size that satisfies (23) and (24). Moreover, Appendix D shows that the sufficient decrease condition (23) provides a necessary condition for the global convergence of subBFGS.

Employing an exact line search is a common strategy to speed up convergence, but it drastically increases the probability of landing on a non-differentiable point (as in Figure 4, left). In order to leverage the fast convergence provided by an exact line search, one must therefore use an optimizer that can handle subgradients, like our subBFGS.

A natural question to ask is whether the optimal step size η^* obtained by an exact line search satisfies the reformulated Wolfe conditions (resp., the standard Wolfe conditions when J is smooth). The answer is no: depending on the choice of c_1, η^* may violate the sufficient decrease condition (23). For the function shown in Figure 8, for instance, we can increase the value of c_1 such that the acceptable interval for the step size excludes η^*. In practice one can set c_1 to a small value, for example, 10^{-4}, to prevent this from happening.

The curvature condition (24), on the other hand, is always satisfied by η^*, as long as $\boldsymbol{p}_t$ is a descent direction (16):

$$\sup_{\boldsymbol{g}' \in J(\boldsymbol{w}_t + \eta^* \boldsymbol{p}_t)} \boldsymbol{g}'^\top \boldsymbol{p}_t = \sup_{g \in \partial \Phi(\eta^*)} g \geq 0 > \sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}_t$$

because $0 \in \partial \, \Phi(\eta^*)$.

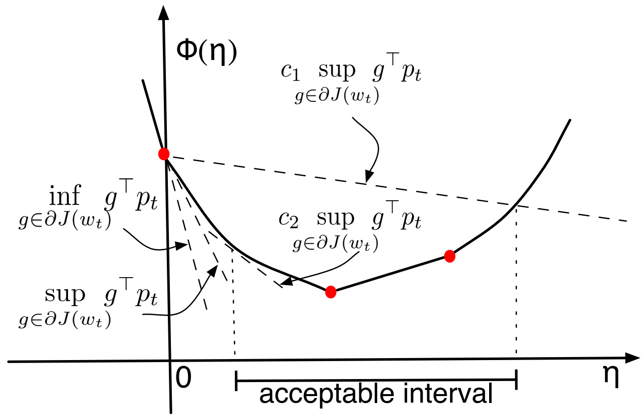


Figure 8: Geometric illustration of the subgradient Wolfe conditions (23) and (24). Solid disks are subdifferentiable points; the slopes of dashed lines are indicated.

3.4 Bounded Spectrum of SubBFGS' Inverse Hessian Estimate

Recall from Section 1 that to ensure positivity of BFGS' estimate $\boldsymbol{B}_t$ of the inverse Hessian, we must have $(\forall t)\ \boldsymbol{s}_t^\top \boldsymbol{y}_t > 0$. Extending this condition to nonsmooth functions, we require

$$(\boldsymbol{w}_{t+1} - \boldsymbol{w}_t)^\top (\boldsymbol{g}_{t+1} - \boldsymbol{g}_t) > 0, \quad \text{where } \boldsymbol{g}_{t+1} \in \partial J(\boldsymbol{w}_{t+1}) \text{ and } \boldsymbol{g}_t \in \partial J(\boldsymbol{w}_t). \tag{25}$$

If J is strongly convex,[5] and $\boldsymbol{w}_{t+1} \neq \boldsymbol{w}_t$, then (25) holds for any choice of $\boldsymbol{g}_{t+1}$ and $\boldsymbol{g}_t$.[6] For general convex functions, $\boldsymbol{g}_{t+1}$ need to be chosen (Line 12 of Algorithm 1) to satisfy (25). The existence of such a subgradient is guaranteed by the convexity of the objective function. To see this, we first use the fact that $\eta_t \boldsymbol{p}_t = \boldsymbol{w}_{t+1} - \boldsymbol{w}_t$ and $\eta_t > 0$ to rewrite (25) as

$$\boldsymbol{p}_t^\top \boldsymbol{g}_{t+1} > \boldsymbol{p}_t^\top \boldsymbol{g}_t, \quad \text{where } \boldsymbol{g}_{t+1} \in \partial J(\boldsymbol{w}_{t+1}) \text{ and } \boldsymbol{g}_t \in \partial J(\boldsymbol{w}_t). \tag{26}$$

It follows from (22) that both sides of inequality (26) are subgradients of $\Phi(\eta)$ at η_t and 0, respectively. The monotonic property of $\partial\Phi(\eta)$ given in Theorem 1 (below) ensures that $\boldsymbol{p}_t^\top \boldsymbol{g}_{t+1}$ is no less than $\boldsymbol{p}_t^\top \boldsymbol{g}_t$ for any choice of $\boldsymbol{g}_{t+1}$ and $\boldsymbol{g}_t$, that is,

$$\inf_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_{t+1})} \boldsymbol{p}_t^\top \boldsymbol{g} \geq \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{p}_t^\top \boldsymbol{g}. \tag{27}$$

This means that the only case where inequality (26) is violated is when both terms of (27) are equal, and

$$\boldsymbol{g}_{t+1} = \operatorname*{arg\,inf}_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_{t+1})} \boldsymbol{g}^\top \boldsymbol{p}_t \quad \text{and} \quad \boldsymbol{g}_t = \operatorname*{arg\,sup}_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}_t,$$

that is, in this case $\boldsymbol{p}_t^\top \boldsymbol{g}_{t+1} = \boldsymbol{p}_t^\top \boldsymbol{g}_t$. To avoid this, we simply need to set $\boldsymbol{g}_{t+1}$ to a different subgradient in $\partial J(\boldsymbol{w}_{t+1})$.

5. If J is strongly convex, then $(\boldsymbol{g}_2 - \boldsymbol{g}_1)^\top (\boldsymbol{w}_2 - \boldsymbol{w}_1) \geq c\,\|\boldsymbol{w}_2 - \boldsymbol{w}_1\|^2$, with $c > 0$, $\boldsymbol{g}_i \in \partial J(\boldsymbol{w}_i)$, $i = 1, 2$.
6. We found empirically that no qualitative difference between using random subgradients versus choosing a particular subgradient when updating the $\boldsymbol{B}_t$ matrix.

Theorem 1 (Hiriart-Urruty and Lemaréchal, 1993, Theorem I.4.2.1)
Let Φ be a one-dimensional convex function on its domain, then $\partial\Phi(\eta)$ is increasing in the sense that $g_1 \leq g_2$ whenever $g_1 \in \partial\Phi(\eta_1)$, $g_2 \in \partial\Phi(\eta_2)$, and $\eta_1 < \eta_2$.

Our convergence analysis for the direction-finding procedure (Algorithm 2) as well as the global convergence proof of subBFGS in Appendix D require the spectrum of $\boldsymbol{B}_t$ to be bounded from above and below by a positive scalar:

$$\exists\,(h, H : 0 < h \leq H < \infty) : (\forall t)\; h \preceq \boldsymbol{B}_t \preceq H. \tag{28}$$

From a theoretical point of view it is difficult to guarantee (28) (Nocedal and Wright, 1999, page 212), but based on the fact that $\boldsymbol{B}_t$ is an approximation to the inverse Hessian $\boldsymbol{H}_t^{-1}$, it is reasonable to expect (28) to be true if

$$(\forall t)\; 1/H \preceq \boldsymbol{H}_t \preceq 1/h.$$

Since BFGS "senses" the Hessian via (6) only through the parameter and gradient displacements $\boldsymbol{s}_t$ and $\boldsymbol{y}_t$, we can translate the bounds on the spectrum of $\boldsymbol{H}_t$ into conditions that only involve $\boldsymbol{s}_t$ and $\boldsymbol{y}_t$:

$$(\forall t)\;\; \frac{\boldsymbol{s}_t^\top \boldsymbol{y}_t}{\boldsymbol{s}_t^\top \boldsymbol{s}_t} \geq \frac{1}{H} \;\;\text{and}\;\; \frac{\boldsymbol{y}_t^\top \boldsymbol{y}_t}{\boldsymbol{s}_t^\top \boldsymbol{y}_t} \leq \frac{1}{h}, \;\;\text{with}\;\; 0 < h \leq H < \infty. \tag{29}$$

This technique is used in Nocedal and Wright (1999, Theorem 8.5). If J is strongly convex[5] and $\boldsymbol{s}_t \neq \boldsymbol{0}$, then there exists an H such that the left inequality in (29) holds. On general convex functions, one can skip BFGS' curvature update if $(\boldsymbol{s}_t^\top \boldsymbol{y}_t / \boldsymbol{s}_t^\top \boldsymbol{s}_t)$ falls below a threshold. To establish the second inequality, we add a fraction of $\boldsymbol{y}_t$ to $\boldsymbol{s}_t$ at Line 14 of Algorithm 1 (though this modification is never actually invoked in our experiments of Section 8, where we set $h = 10^{-8}$).

3.5 Limited-Memory Subgradient BFGS

It is straightforward to implement an LBFGS variant of our subBFGS algorithm: we simply modify Algorithms 1 and 2 to compute all products between $\boldsymbol{B}_t$ and a vector by means of the standard LBFGS matrix-free scheme (Nocedal and Wright, 1999, Algorithm 9.1). We call the resulting algorithm subLBFGS.

3.6 Convergence of Subgradient (L)BFGS

In Section 3.4 we have shown that the spectrum of subBFGS' inverse Hessian estimate is bounded. From this and other technical assumptions, we prove in Appendix D that subBFGS is globally convergent in objective function value, that is, $J(\boldsymbol{w}) \to \inf_{\boldsymbol{w}} J(\boldsymbol{w})$. Moreover, in Appendix E we show that subBFGS converges for all counterexamples we could find in the literature used to illustrate the non-convergence of existing optimization methods on nonsmooth problems.

We have also examined the convergence of subLBFGS empirically. In most of our experiments of Section 8, we observe that after an initial transient, subLBFGS observes a period of linear convergence, until close to the optimum it exhibits superlinear convergence behavior. This is illustrated

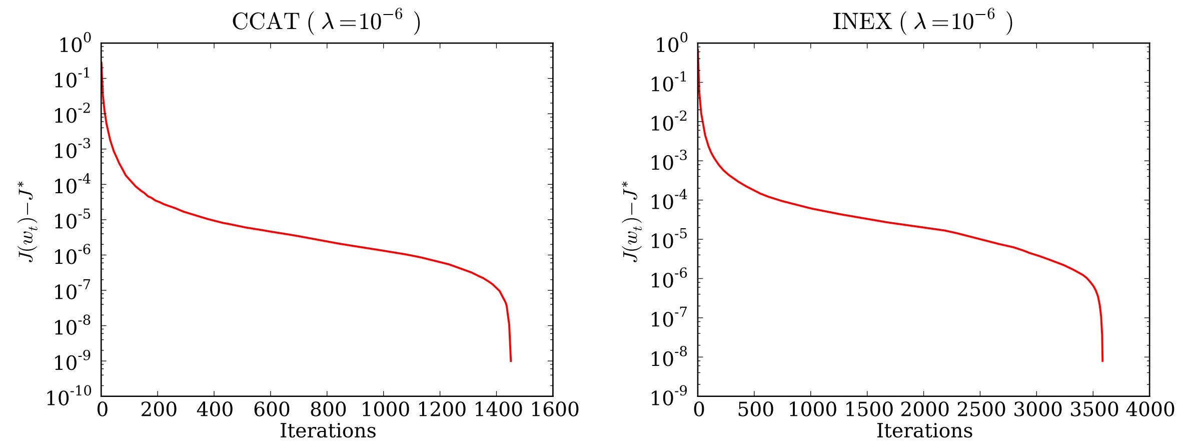


Figure 9: Convergence of subLBFGS in objective function value on sample L_2-regularized risk minimization problems with binary (left) and multiclass (right) hinge losses.

in Figure 9, where we plot (on a log scale) the excess objective function value $J(\boldsymbol{w}_t)$ over its "optimum" J^*[7] against the iteration number in two typical runs. The same kind of convergence behavior was observed by Lewis and Overton (2008a, Figure 5.7), who applied the classical BFGS algorithm with a specially designed line search to nonsmooth functions. They caution that the apparent superlinear convergence may be an artifact caused by the inaccuracy of the estimated optimal value of the objective.

4. SubBFGS for L_2-Regularized Binary Hinge Loss

Many machine learning algorithms can be viewed as minimizing the L_2-regularized risk

$$J(\boldsymbol{w}) \;:=\; \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \;+\; \frac{1}{n}\sum_{i=1}^{n} l(\boldsymbol{x}_i, z_i, \boldsymbol{w}), \tag{30}$$

where $\lambda > 0$ is a regularization constant, $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ are the input features, $z_i \in \mathcal{Z} \subseteq \mathbb{Z}$ the corresponding labels, and the loss l is a non-negative convex function of $\boldsymbol{w}$ which measures the discrepancy between z_i and the predictions arising from using $\boldsymbol{w}$. A loss function commonly used for binary classification is the binary hinge loss

$$l(\boldsymbol{x}, z, \boldsymbol{w}) \;:=\; \max(0, 1 - z\,\boldsymbol{w}^\top \boldsymbol{x}), \tag{31}$$

where $z \in \{\pm 1\}$. L_2-regularized risk minimization with the binary hinge loss is a convex but nonsmooth optimization problem; in this section we show how subBFGS (Algorithm 1) can be applied to this problem.

7. Estimated empirically by running subLBFGS for 10^4 seconds, or until the relative improvement over 5 iterations was less than 10^{-8}.

Let $\mathcal{E}$, $\mathcal{M}$, and $\mathcal{W}$ index the set of points which are in error, on the margin, and well-classified, respectively:

$$\mathcal{E} := \{i \in \{1, 2, \ldots, n\} : 1 - z_i \boldsymbol{w}^\top \boldsymbol{x}_i > 0\},$$
$$\mathcal{M} := \{i \in \{1, 2, \ldots, n\} : 1 - z_i \boldsymbol{w}^\top \boldsymbol{x}_i = 0\},$$
$$\mathcal{W} := \{i \in \{1, 2, \ldots, n\} : 1 - z_i \boldsymbol{w}^\top \boldsymbol{x}_i < 0\}.$$

Differentiating (30) after plugging in (31) then yields

$$\partial J(\boldsymbol{w}) \;=\; \lambda \boldsymbol{w} - \frac{1}{n}\sum_{i=1}^{n} \beta_i z_i \boldsymbol{x}_i \;=\; \bar{\boldsymbol{w}} - \frac{1}{n}\sum_{i\in\mathcal{M}} \beta_i z_i \boldsymbol{x}_i, \tag{32}$$

$$\text{where}\quad \bar{\boldsymbol{w}} := \lambda \boldsymbol{w} - \frac{1}{n}\sum_{i\in\mathcal{E}} z_i \boldsymbol{x}_i \quad\text{and}\quad \beta_i := \begin{cases} 1 & \text{if } i \in \mathcal{E}, \\ [0,1] & \text{if } i \in \mathcal{M}, \\ 0 & \text{if } i \in \mathcal{W}. \end{cases}$$

4.1 Efficient Oracle for the Direction-Finding Method

Recall that subBFGS requires an oracle that provides $\arg\sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}$ for a given direction $\boldsymbol{p}$. For L_2-regularized risk minimization with the binary hinge loss we can implement such an oracle at a computational cost of $O(d\,|\,\mathcal{M}_t\,|)$, where d is the dimensionality of $\boldsymbol{p}$ and $|\,\mathcal{M}_t\,|$ the number of current margin points, which is normally much less than n. Towards this end, we use (32) to obtain

$$\begin{aligned} \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p} \;&=\; \sup_{\beta_i, i\in\mathcal{M}_t} \left(\bar{\boldsymbol{w}}_t - \frac{1}{n}\sum_{i\in\mathcal{M}_t} \beta_i z_i \boldsymbol{x}_i\right)^{\!\top} \boldsymbol{p} \\ &=\; \bar{\boldsymbol{w}}_t^\top \boldsymbol{p} \;-\; \frac{1}{n}\sum_{i\in\mathcal{M}_t} \inf_{\beta_i\in[0,1]} (\beta_i z_i \boldsymbol{x}_i^\top \boldsymbol{p}). \end{aligned} \tag{33}$$

Since for a given $\boldsymbol{p}$ the first term of the right-hand side of (33) is a constant, the supremum is attained when we set $\beta_i\ \forall i \in \mathcal{M}_t$ via the following strategy:

$$\beta_i := \begin{cases} 0 & \text{if } z_i \boldsymbol{x}_i^\top \boldsymbol{p}_t \;\geq\; 0, \\ 1 & \text{if } z_i \boldsymbol{x}_i^\top \boldsymbol{p}_t \;<\; 0. \end{cases}$$

4.2 Implementing the Line Search

The one-dimensional convex function $\Phi(\eta) := J(\boldsymbol{w} + \eta \boldsymbol{p})$ (Figure 10, left) obtained by restricting (30) to a line can be evaluated efficiently. To see this, rewrite (30) as

$$J(\boldsymbol{w}) \;:=\; \frac{\lambda}{2}\,\|\boldsymbol{w}\|^2 \;+\; \frac{1}{n}\,\mathbf{1}^\top \max(\mathbf{0},\; \mathbf{1} - \boldsymbol{z}\cdot \boldsymbol{X}\boldsymbol{w}), \tag{34}$$

where $\mathbf{0}$ and $\mathbf{1}$ are column vectors of zeros and ones, respectively, $\cdot$ denotes the Hadamard (component-wise) product, and $\boldsymbol{z} \in \mathbb{R}^n$ collects correct labels corresponding to each row of data in $\boldsymbol{X} := [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n]^\top \in \mathbb{R}^{n\times d}$. Given a search direction $\boldsymbol{p}$ at a point $\boldsymbol{w}$, (34) allows us to write

$$\Phi(\eta) = \frac{\lambda}{2}\,\|\boldsymbol{w}\|^2 + \lambda\,\eta\,\boldsymbol{w}^\top \boldsymbol{p} \;+\; \frac{\lambda\,\eta^2}{2}\,\|\boldsymbol{p}\|^2 \;+\; \frac{1}{n}\,\mathbf{1}^\top \max\left[0,\; (\mathbf{1} - (\boldsymbol{f} + \eta\,\Delta \boldsymbol{f}))\right], \tag{35}$$

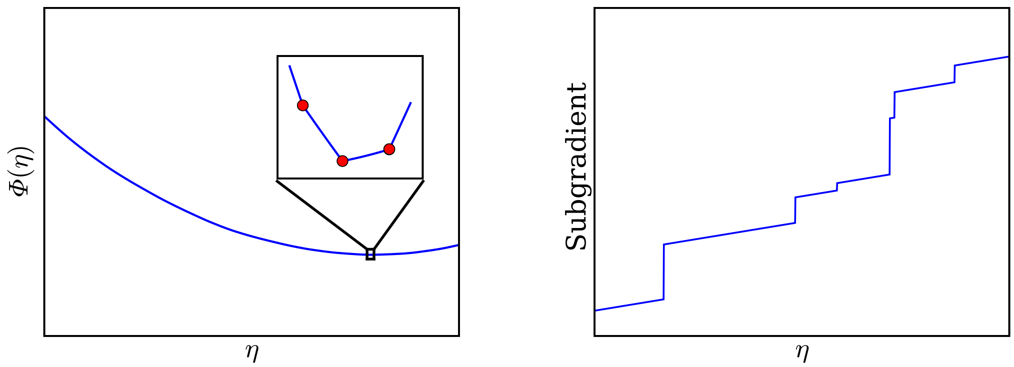


Figure 10: Left: Piecewise quadratic convex function Φ of step size η; solid disks in the zoomed inset are subdifferentiable points. Right: The subgradient of $\Phi(\eta)$ increases monotonically with η, and jumps discontinuously at subdifferentiable points.

where $\boldsymbol{f} := \boldsymbol{z} \cdot \boldsymbol{X}\boldsymbol{w}$ and $\Delta\boldsymbol{f} := \boldsymbol{z} \cdot \boldsymbol{X}\boldsymbol{p}$. Differentiating (35) with respect to η gives the subdifferential of Φ:

$$\partial\,\Phi(\eta) = \lambda \boldsymbol{w}^\top \boldsymbol{p} + \eta\lambda\|\boldsymbol{p}\|^2 - \frac{1}{n}\boldsymbol{\delta}(\eta)^\top \Delta\boldsymbol{f}, \tag{36}$$

where $\boldsymbol{\delta} : \mathbb{R} \to \mathbb{R}^n$ outputs a column vector $[\delta_1(\eta), \delta_2(\eta), \cdots, \delta_n(\eta)]^\top$ with

$$\delta_i(\eta) := \begin{cases} 1 & \text{if} \quad f_i + \eta\,\Delta f_i \; < \; 1, \\ [0,1] & \text{if} \quad f_i + \eta\,\Delta f_i \; = \; 1, \\ 0 & \text{if} \quad f_i + \eta\,\Delta f_i \; > \; 1. \end{cases} \tag{37}$$

We cache $\boldsymbol{f}$ and $\Delta\boldsymbol{f}$, expending $O(nd)$ computational effort and using $O(n)$ storage. We also cache the scalars $\frac{\lambda}{2}\|\boldsymbol{w}\|^2$, $\lambda\,\boldsymbol{w}^\top\boldsymbol{p}$, and $\frac{\lambda}{2}\|\boldsymbol{p}\|^2$, each of which requires $O(d)$ work. The evaluation of $\boldsymbol{1} - (\boldsymbol{f} + \eta\,\Delta\boldsymbol{f})$, $\boldsymbol{\delta}(\eta)$, and the inner products in the final terms of (35) and (36) all take $O(n)$ effort. Given the cached terms, all other terms in (35) can be computed in constant time, thus reducing the cost of evaluating $\Phi(\eta)$ (resp., its subgradient) to $O(n)$. Furthermore, from (37) we see that $\Phi(\eta)$ is differentiable everywhere except at

$$\eta_i := (1 - f_i)/\Delta f_i \;\text{ with }\; \Delta f_i \neq 0, \tag{38}$$

where it becomes subdifferentiable. At these points an element of the indicator vector (37) changes from 0 to 1 or vice versa (causing the subgradient to jump, as shown in Figure 10, right); otherwise $\boldsymbol{\delta}(\eta)$ remains constant. Using this property of $\boldsymbol{\delta}(\eta)$, we can update the last term of (36) in constant time when passing a hinge point (Line 25 of Algorithm 3). We are now in a position to introduce an exact line search which takes advantage of this scheme.

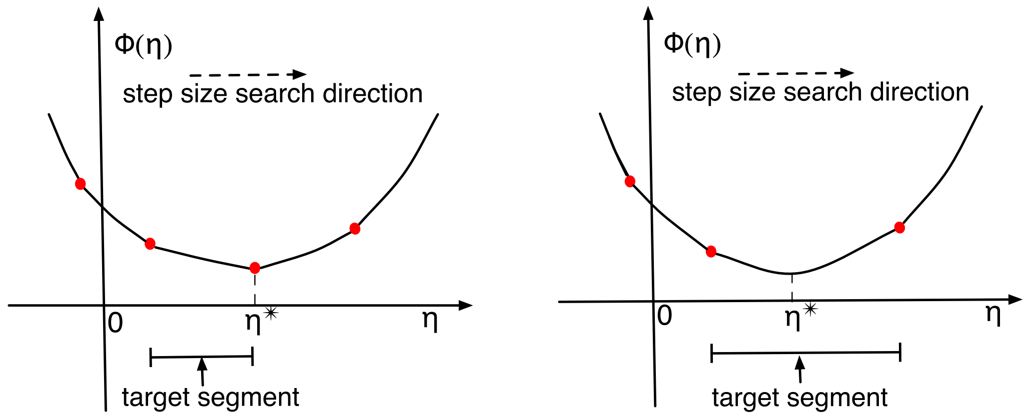


Figure 11: Nonsmooth convex function Φ of step size η. Solid disks are subdifferentiable points; the optimal step η^* either falls on such a point (left), or lies between two such points (right).

4.2.1 EXACT LINE SEARCH

Given a direction $\boldsymbol{p}$, exact line search finds the optimal step size $\eta^* := \operatorname{argmin}_{\eta \geq 0} \Phi(\eta)$ that satisfies $0 \in \partial\,\Phi(\eta^*)$, or equivalently

$$\inf \partial\,\Phi(\eta^*) \leq 0 \leq \sup \partial\,\Phi(\eta^*).$$

By Theorem 1, $\sup \partial\,\Phi(\eta)$ is monotonically increasing with η. Based on this property, our algorithm first builds a list of all possible subdifferentiable points and $\eta = 0$, sorted by non-descending value of η (Lines 4–5 of Algorithm 3). Then, it starts with $\eta = 0$, and walks through the sorted list until it locates the "target segment", an interval $[\eta_a, \eta_b]$ between two subdifferential points with $\sup \partial\,\Phi(\eta_a) \leq 0$ and $\sup \partial\,\Phi(\eta_b) \geq 0$. We now know that the optimal step size either coincides with η_b (Figure 11, left), or lies in (η_a, η_b) (Figure 11, right). If η^* lies in the smooth interval (η_a, η_b), then setting (36) to zero gives

$$\eta^* = \frac{\boldsymbol{\delta}(\eta')^\top \boldsymbol{\Delta f}/n - \lambda\,\boldsymbol{w}^\top \boldsymbol{p}}{\lambda\,\|\boldsymbol{p}\|^2}, \quad \forall \eta' \in (\eta_a, \eta_b). \tag{39}$$

Otherwise, $\eta^* = \eta_b$. See Algorithm 3 for the detailed implementation.

5. Segmenting the Pointwise Maximum of 1-D Linear Functions

The line search of Algorithm 3 requires a vector $\boldsymbol{\eta}$ listing the subdifferentiable points along the line $\boldsymbol{w} + \eta\boldsymbol{p}$, and sorts it in non-descending order (Line 5). For an objective function like (30) whose nonsmooth component is just a sum of hinge losses (31), this vector is very easy to compute (cf. (38)). In order to apply our line search approach to multiclass and multilabel losses, however, we must solve a more general problem: we need to efficiently find the subdifferentiable points of a

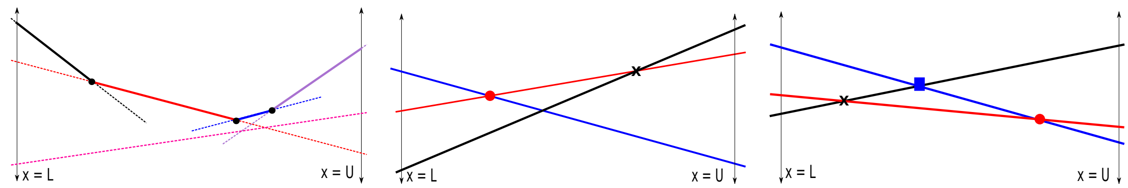
Algorithm 3 Exact Line Search for L_2-Regularized Binary Hinge Loss

input $\boldsymbol{w}, \boldsymbol{p}, \lambda, \boldsymbol{f}$, and $\Delta\boldsymbol{f}$ as in (35)
output optimal step size
$h = \lambda\|\boldsymbol{p}\|^2,\ j := 1$
$\boldsymbol{\eta} := [(\mathbf{1} - \boldsymbol{f})./\Delta\boldsymbol{f}, 0]$ (vector of subdifferentiable points & zero)
$\boldsymbol{\pi} = \mathtt{argsort}(\boldsymbol{\eta})$ (indices sorted by non-descending value of $\boldsymbol{\eta}$)
while $\eta_{\pi_j} \leq 0$ **do**
  $j := j + 1$
end while
$\eta := \eta_{\pi_j}/2$
for $i := 1$ to $\boldsymbol{f}.\mathtt{size}$ **do**
  $\delta_i := \begin{cases} 1 & \text{if } f_i + \eta\,\Delta f_i < 1 \\ 0 & \text{otherwise} \end{cases}$ (value of $\boldsymbol{\delta}(\eta)$ (37) for any $\eta \in (0, \eta_{\pi_j})$)
end for
$\varrho := \boldsymbol{\delta}^\top \Delta\boldsymbol{f}/n - \lambda\boldsymbol{w}^\top\boldsymbol{p}$
$\eta := 0,\ \varrho' := 0$
$g := -\varrho$ (value of $\sup\partial\,\Phi(0)$)
while $g < 0$ **do**
  $\varrho' := \varrho$
  if $j > \boldsymbol{\pi}.\mathtt{size}$ **then**
    $\eta := \infty$ (no more subdifferentiable points)
    break
  else
    $\eta := \eta_{\pi_j}$
  end if
  repeat
    $\varrho := \begin{cases} \varrho - \Delta f_{\pi_j}/n & \text{if } \delta_{\pi_j} = 1 \\ \varrho + \Delta f_{\pi_j}/n & \text{otherwise} \end{cases}$ (move to next subdifferentiable point and update ϱ accordingly)
    $j := j + 1$
  until $\eta_{\pi_j} \neq \eta_{\pi_{j-1}}$ and $j \leq \boldsymbol{\pi}.\mathtt{size}$
  $g := \eta h - \varrho$ (value of $\sup\partial\,\Phi(\eta_{\pi_{j-1}})$)
end while
return $\min(\eta,\ \varrho'/h)$ (cf. equation 39)

one-dimensional piecewise linear function $\varrho : \mathbb{R} \to \mathbb{R}$ defined to be the pointwise maximum of r lines:

$$\varrho(\eta) = \max_{1 \leq p \leq r} (b_p + \eta\, a_p), \tag{40}$$

where a_p and b_p denote the slope and offset of the p^{th} line, respectively. Clearly, ϱ is convex since it is the pointwise maximum of linear functions (Boyd and Vandenberghe, 2004, Section 3.2.3), see Figure 12(a). The difficulty here is that although ϱ consists of at most r line segments bounded by at most $r - 1$ subdifferentiable points, there are $r(r-1)/2$ candidates for these points, namely all intersections between any two of the r lines. A naive algorithm to find the subdifferentiable points of ϱ would therefore take $O(r^2)$ time. In what follows, however, we show how this can be done in just $O(r \log r)$ time. In Section 6 we will then use this technique (Algorithm 4) to perform efficient exact line search in the multiclass and multilabel settings.



(a) Pointwise maximum of lines (b) Case 1 (c) Case 2

Figure 12: (a) Convex piecewise linear function defined as the maximum of 5 lines, but comprising only 4 active line segments (bold) separated by 3 subdifferentiable points (black dots). (b, c) Two cases encountered by our algorithm: (b) The new intersection (black cross) lies to the right of the previous one (red dot) and is therefore pushed onto the stack; (c) The new intersection lies to the left of the previous one. In this case the latter is popped from the stack, and a third intersection (blue square) is computed and pushed onto it.

Algorithm 4 Segmenting a Pointwise Maximum of 1-D Linear Functions

input vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ of slopes and offsets
    lower bound L, upper bound U, with $0 \leq L < U < \infty$
output sorted stack of subdifferentiable points $\boldsymbol{\eta}$
    and corresponding active line indices $\boldsymbol{\xi}$
$\mathbf{y} := \boldsymbol{b} + L\boldsymbol{a}$
$\boldsymbol{\pi} := \mathtt{argsort}(-\mathbf{y})$ (indices sorted by non-ascending value of $\mathbf{y}$)
$S.\mathtt{push}\ (L, \pi_1)$ (initialize stack)
for $q := 2$ $\mathtt{to}$ $\mathbf{y}.\mathtt{size}$ **do**
  while not $S.\mathtt{empty}$ **do**
    $(\eta, \xi) := S.\mathtt{top}$
    $\eta' := \dfrac{b_{\pi_q} - b_\xi}{a_\xi - a_{\pi_q}}$ (intersection of two lines)
    if $L < \eta' \leq \eta$ or $(\eta' = L$ and $a_{\pi_q} > a_\xi)$ **then**
      $S.\mathtt{pop}$ (cf. Figure 12(c))
    else
      break
    end if
  end while
  if $L < \eta' \leq U$ or $(\eta' = L$ and $a_{\pi_q} > a_\xi)$ **then**
    $S.\mathtt{push}\ (\eta', \pi_q)$ (cf. Figure 12(b))
  end if
end for
return S

We begin by specifying an interval $[L, U]$ $(0 \leq L < U < \infty)$ in which to find the subdifferentiable points of ϱ, and set $\mathbf{y} := \boldsymbol{b} + L\boldsymbol{a}$, where $\boldsymbol{a} = [a_1, a_2, \cdots, a_r]$ and $\boldsymbol{b} = [b_1, b_2, \cdots, b_r]$. In other words, $\mathbf{y}$ contains the intersections of the r lines defining $\varrho(\eta)$ with the vertical line $\eta = L$. Let $\boldsymbol{\pi}$ denote the permutation that sorts $\mathbf{y}$ in non-ascending order, that is, $p < q \implies y_{\pi_p} \geq y_{\pi_q}$, and let $\varrho^{(q)}$ be the function obtained by considering only the top $q \leq r$ lines at $\eta = L$, that is, the first q lines in $\boldsymbol{\pi}$:

$$\varrho^{(q)}(\eta) = \max_{1 \leq p \leq q} (b_{\pi_p} + \eta\, a_{\pi_p}). \tag{41}$$

It is clear that $\varrho^{(r)} = \varrho$. Let $\boldsymbol{\eta}$ contain all $q' \leq q-1$ subdifferentiable points of $\varrho^{(q)}$ in $[L, U]$ in ascending order, and $\boldsymbol{\xi}$ the indices of the corresponding *active* lines, that is, the maximum in (41) is attained for line ξ_{j-1} over the interval $[\eta_{j-1}, \eta_j]$: $\xi_{j-1} := \pi_{p^*}$, where $p^* = \mathrm{argmax}_{1 \leq p \leq q}(b_{\pi_p} + \eta a_{\pi_p})$ for $\eta \in [\eta_{j-1}, \eta_j]$, and lines ξ_{j-1} and ξ_j intersect at η_j.

Initially we set $\eta_0 := L$ and $\xi_0 := \pi_1$, the leftmost bold segment in Figure 12(a). Algorithm 4 goes through lines in $\boldsymbol{\pi}$ sequentially, and maintains a Last-In-First-Out stack S which at the end of the q^{th} iteration consists of the tuples

$$(\eta_0, \xi_0), (\eta_1, \xi_1), \ldots, (\eta_{q'}, \xi_{q'})$$

in order of ascending η_i, with $(\eta_{q'}, \xi_{q'})$ at the top. After r iterations S contains a sorted list of all subdifferentiable points (and the corresponding active lines) of $\varrho = \varrho^{(r)}$ in $[L, U]$, as required by our line searches.

In iteration $q+1$ Algorithm 4 examines the intersection η' between lines $\xi_{q'}$ and π_{q+1}: If $\eta' > U$, line π_{q+1} is irrelevant, and we proceed to the next iteration. If $\eta_{q'} < \eta' \leq U$ as in Figure 12(b), then line π_{q+1} is becoming active at η', and we simply push (η', π_{q+1}) onto the stack. If $\eta' \leq \eta_{q'}$ as in Figure 12(c), on the other hand, then line π_{q+1} dominates line $\xi_{q'}$ over the interval (η', ∞) and hence over $(\eta_{q'}, U] \subset (\eta', \infty)$, so we pop $(\eta_{q'}, \xi_{q'})$ from the stack (deactivating line $\xi_{q'}$), decrement q', and repeat the comparison.

Theorem 2 *The total running time of Algorithm 4 is $O(r \log r)$.*

Proof Computing intersections of lines as well as pushing and popping from the stack require $O(1)$ time. Each of the r lines can be pushed onto and popped from the stack at most once; amortized over r iterations the running time is therefore $O(r)$. The time complexity of Algorithm 4 is thus dominated by the initial sorting of $\mathbf{y}$ (i.e., the computation of $\boldsymbol{\pi}$), which takes $O(r \log r)$ time. ■

6. SubBFGS for Multiclass and Multilabel Hinge Losses

We now use the algorithm developed in Section 5 to generalize the subBFGS method of Section 4 to the multiclass and multilabel settings with finite label set $\mathcal{Z}$. We assume that given a feature vector $\boldsymbol{x}$ our classifier predicts the label

$$z^* = \underset{z \in \mathcal{Z}}{\mathrm{argmax}}\, f(\boldsymbol{w}, \boldsymbol{x}, z),$$

where f is a linear function of $\boldsymbol{w}$, that is, $f(\boldsymbol{w}, \boldsymbol{x}, z) = \boldsymbol{w}^\top \phi(\boldsymbol{x}, z)$ for some feature map $\phi(\boldsymbol{x}, z)$.

6.1 Multiclass Hinge Loss

A variety of multiclass hinge losses have been proposed in the literature that generalize the binary hinge loss, and enforce a margin of separation between the true label z_i and every other label. We focus on the following rather general variant (Taskar et al., 2004):[8]

$$l(\boldsymbol{x}_i, z_i, \boldsymbol{w}) \;:=\; \max_{z \in \mathcal{Z}} [\Delta(z, z_i) + f(\boldsymbol{w}, \boldsymbol{x}_i, z) - f(\boldsymbol{w}, \boldsymbol{x}_i, z_i)], \tag{42}$$

where $\Delta(z, z_i) \geq 0$ is the *label loss* specifying the margin required between labels z and z_i. For instance, a uniform margin of separation is achieved by setting $\Delta(z, z') := \tau > 0 \; \forall z \neq z'$ (Crammer and Singer, 2003a). By requiring that $\forall z \in \mathcal{Z} : \Delta(z, z) = 0$ we ensure that (42) always remains non-negative. Adapting (30) to the multiclass hinge loss (42) we obtain

$$J(\boldsymbol{w}) \;:=\; \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \max_{z \in \mathcal{Z}} [\Delta(z, z_i) + f(\boldsymbol{w}, \boldsymbol{x}_i, z) - f(\boldsymbol{w}, \boldsymbol{x}_i, z_i)]. \tag{43}$$

For a given $\boldsymbol{w}$, consider the set

$$\mathcal{Z}_i^* := \operatorname*{argmax}_{z \in \mathcal{Z}} [\Delta(z, z_i) + f(\boldsymbol{w}, \boldsymbol{x}_i, z) - f(\boldsymbol{w}, \boldsymbol{x}_i, z_i)]$$

of maximum-loss labels (possibly more than one) for the i^{th} training instance. Since $f(\boldsymbol{w}, \boldsymbol{x}, z) = \boldsymbol{w}^\top \phi(\boldsymbol{x}, z)$, the subdifferential of (43) can then be written as

$$\partial J(\boldsymbol{w}) \;=\; \lambda \boldsymbol{w} + \frac{1}{n}\sum_{i=1}^{n}\sum_{z \in \mathcal{Z}} \beta_{i,z}\, \phi(\boldsymbol{x}_i, z) \tag{44}$$

$$\text{with} \quad \beta_{i,z} \;=\; \left\{ \begin{array}{rl} [0,1] & \text{if } z \in \mathcal{Z}_i^* \\ 0 & \text{otherwise} \end{array} \right\} - \delta_{z,z_i} \;\text{ s.t. } \sum_{z \in \mathcal{Z}} \beta_{i,z} = 0, \tag{45}$$

where δ is the Kronecker delta: $\delta_{a,b} = 1$ if $a = b$, and 0 otherwise.[9]

6.2 Efficient Multiclass Direction-Finding Oracle

For L_2-regularized risk minimization with multiclass hinge loss, we can use a similar scheme as described in Section 4.1 to implement an efficient oracle that provides $\arg\sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w})} \boldsymbol{g}^\top \boldsymbol{p}$ for the direction-finding procedure (Algorithm 2). Using (44), we can write

$$\sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w})} \boldsymbol{g}^\top \boldsymbol{p} \;=\; \lambda \boldsymbol{w}^\top \boldsymbol{p} \;+\; \frac{1}{n}\sum_{i=1}^{n}\sum_{z \in \mathcal{Z}} \sup_{\beta_{i,z}} \Big(\beta_{i,z}\, \phi(\boldsymbol{x}_i, z)^\top \boldsymbol{p} \Big). \tag{46}$$

The supremum in (46) is attained when we pick, from the choices offered by (45),

$$\beta_{i,z} \;:=\; \delta_{z,z_i^*} - \delta_{z,z_i}, \quad \text{where} \quad z_i^* \;:=\; \operatorname*{argmax}_{z \in \mathcal{Z}_i^*} \phi(\boldsymbol{x}_i, z)^\top \boldsymbol{p}.$$

8. Our algorithm can also deal with the slack-rescaled variant of Tsochantaridis et al. (2005).

9. Let $l_i^* := \max_{z \neq z_i} [\Delta(z, z_i) + f(\boldsymbol{w}, \boldsymbol{x}_i, z) - f(\boldsymbol{w}, \boldsymbol{x}_i, z_i)]$. Definition (45) allows the following values of $\beta_{i,z}$:

$$\left\{ \begin{array}{c|ccc} & z = z_i & z \in \mathcal{Z}_i^* \backslash \{z_i\} & \text{otherwise} \\ \hline l_i^* < 0 & 0 & 0 & 0 \\ l_i^* = 0 & [-1, 0] & [0, 1] & 0 \\ l_i^* > 0 & -1 & [0, 1] & 0 \end{array} \right\} \;\text{ s.t. } \sum_{z \in \mathcal{Z}} \beta_{i,z} = 0.$$

6.3 Implementing the Multiclass Line Search

Let $\Phi(\eta) := J(\boldsymbol{w} + \eta \boldsymbol{p})$ be the one-dimensional convex function obtained by restricting (43) to a line along direction $\boldsymbol{p}$. Letting $\varrho_i(\eta) := l(\boldsymbol{x}_i, z_i, \boldsymbol{w} + \eta \boldsymbol{p})$, we can write

$$\Phi(\eta) \;=\; \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \lambda\eta\boldsymbol{w}^\top\boldsymbol{p} + \frac{\lambda\eta^2}{2}\|\boldsymbol{p}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\varrho_i(\eta). \tag{47}$$

Each $\varrho_i(\eta)$ is a piecewise linear convex function. To see this, observe that

$$f(\boldsymbol{w} + \eta\boldsymbol{p}, \boldsymbol{x}, z) := (\boldsymbol{w} + \eta\boldsymbol{p})^\top \phi(\boldsymbol{x}, z) = f(\boldsymbol{w}, \boldsymbol{x}, z) + \eta f(\boldsymbol{p}, \boldsymbol{x}, z)$$

and hence

$$\varrho_i(\eta) \;:=\; \max_{z\in\mathcal{Z}} [\underbrace{\Delta(z, z_i) + f(\boldsymbol{w}, \boldsymbol{x}_i, z) - f(\boldsymbol{w}, \boldsymbol{x}_i, z_i)}_{=:\, b_z^{(i)}} + \eta\underbrace{(f(\boldsymbol{p}, \boldsymbol{x}_i, z) - f(\boldsymbol{p}, \boldsymbol{x}_i, z_i))}_{=:\, a_z^{(i)}}], \tag{48}$$

which has the functional form of (40) with $r = |\mathcal{Z}|$. Algorithm 4 can therefore be used to compute a sorted vector $\boldsymbol{\eta}^{(i)}$ of all subdifferentiable points of $\varrho_i(\eta)$ and corresponding active lines $\boldsymbol{\xi}^{(i)}$ in the interval $[0, \infty)$ in $O(|\mathcal{Z}| \log |\mathcal{Z}|)$ time. With some abuse of notation, we now have

$$\eta \in [\eta_j^{(i)}, \eta_{j+1}^{(i)}] \;\Longrightarrow\; \varrho_i(\eta) \;=\; b_{\xi_j^{(i)}} + \eta\, a_{\xi_j^{(i)}}. \tag{49}$$

The first three terms of (47) are constant, linear, and quadratic (with non-negative coefficient) in η, respectively. The remaining sum of piecewise linear convex functions $\varrho_i(\eta)$ is also piecewise linear and convex, and so $\Phi(\eta)$ is a piecewise quadratic convex function.

6.3.1 Exact Multiclass Line Search

Our exact line search employs a similar two-stage strategy as discussed in Section 4.2.1 for locating its minimum $\eta^* := \operatorname{argmin}_{\eta>0} \Phi(\eta)$: we first find the first *subdifferentiable* point $\check{\eta}$ past the minimum, then locate η^* within the differentiable region to its left. We precompute and cache a vector $\boldsymbol{a}^{(i)}$ of all the slopes $a_z^{(i)}$ (offsets $b_z^{(i)}$ are not needed), the subdifferentiable points $\boldsymbol{\eta}^{(i)}$ (sorted in ascending order via Algorithm 4), and the corresponding indices $\boldsymbol{\xi}^{(i)}$ of active lines of ϱ_i for all training instances i, as well as $\|\boldsymbol{w}\|^2$, $\boldsymbol{w}^\top\boldsymbol{p}$, and $\lambda\|\boldsymbol{p}\|^2$.

Since $\Phi(\eta)$ is convex, any point $\eta < \eta^*$ cannot have a non-negative subgradient.[10] The first subdifferentiable point $\check{\eta} \geq \eta^*$ therefore obeys

$$\begin{aligned}\check{\eta} \;&:=\; \min \eta \in \{\boldsymbol{\eta}^{(i)},\, i = 1, 2, \ldots, n\} : \eta \geq \eta^* \\ &=\; \min \eta \in \{\boldsymbol{\eta}^{(i)},\, i = 1, 2, \ldots, n\} : \sup \partial\,\Phi(\eta) \geq 0.\end{aligned} \tag{50}$$

We solve (50) via a simple linear search: Starting from $\eta = 0$, we walk from one subdifferentiable point to the next until $\sup \partial\,\Phi(\eta) \geq 0$. To perform this walk efficiently, define a vector $\boldsymbol{\psi} \in \mathbb{N}^n$ of indices into the sorted vector $\boldsymbol{\eta}^{(i)}$ *resp.* $\boldsymbol{\xi}^{(i)}$; initially $\boldsymbol{\psi} := \mathbf{0}$, indicating that $(\forall i)\ \eta_0^{(i)} = 0$. Given the current index vector $\boldsymbol{\psi}$, the next subdifferentiable point is then

$$\eta' := \eta^{(i')}_{(\psi_{i'}+1)}, \quad \text{where } \; i' = \operatorname*{argmin}_{1\leq i\leq n} \eta^{(i)}_{(\psi_i+1)}; \tag{51}$$

10. If $\Phi(\eta)$ has a flat optimal region, we define η^* to be the infimum of that region.

Algorithm 5 Exact Line Search for L_2-Regularized Multiclass Hinge Loss

1: **input** base point $\boldsymbol{w}$, descent direction $\boldsymbol{p}$, regularization parameter λ, vector $\boldsymbol{a}$ of all slopes as defined in (48), for each training instance i: sorted stack S_i of subdifferentiable points and active lines, as produced by Algorithm 4
2: **output** optimal step size
3: $\boldsymbol{a} := \boldsymbol{a}/n,\ h := \lambda\|\boldsymbol{p}\|^2$
4: $\varrho := \lambda \boldsymbol{w}^\top \boldsymbol{p}$
5: **for** $i := 1$ to n **do**
6:     **while** not S_i.`empty` **do**
7:         R_i.`push` S_i.`pop` (reverse the stacks)
8:     **end while**
9:     $(\cdot, \xi_i) := R_i$.`pop`
10:     $\varrho := \varrho + a_{\xi_i}$
11: **end for**
12: $\eta := 0,\ \varrho' = 0$
13: $g := \varrho$ (value of $\sup \partial\,\Phi(0)$)
14: **while** $g < 0$ **do**
15:     $\varrho' := \varrho$
16:     **if** $\forall i : R_i$.`empty` **then**
17:         $\eta := \infty$ (no more subdifferentiable points)
18:         **break**
19:     **end if**
20:     $\mathcal{I} := \operatorname{argmin}_{1 \le i \le n}\ \eta' :\ (\eta', \cdot) = R_i$.`top` (find the next subdifferentiable point)
21:     $\varrho := \varrho - \sum_{i \in \mathcal{I}} a_{\xi_i}$
22:     $\Xi := \{\xi_i : (\eta, \xi_i) := R_i$.`pop`$,\ \ i \in \mathcal{I}\}$
23:     $\varrho := \varrho + \sum_{\xi_i \in \Xi} a_{\xi_i}$
24:     $g := \varrho + \eta\, h$ (value of $\sup \partial \Phi(\eta)$)
25: **end while**
26: **return** $\min(\eta,\ -\varrho'/h)$

the step is completed by incrementing $\psi_{i'}$, that is, $\psi_{i'} := \psi_{i'} + 1$ so as to remove $\eta^{(i')}_{\psi_{i'}}$ from future consideration.[11] Note that computing the argmin in (51) takes $O(\log n)$ time (e.g., using a priority queue). Inserting (49) into (47) and differentiating, we find that

$$\sup\,\partial\,\Phi(\eta') \;=\; \lambda \boldsymbol{w}^\top \boldsymbol{p} + \lambda\eta'\|\boldsymbol{p}\|^2 + \frac{1}{n}\sum_{i=1}^{n} a_{\xi^{(i)}_{\psi_i}}. \tag{52}$$

The key observation here is that after the initial calculation of $\sup \partial\,\Phi(0) = \lambda \boldsymbol{w}^\top \boldsymbol{p} + \frac{1}{n}\sum_{i=1}^{n} a_{\xi^{(i)}_0}$ for $\eta = 0$, the sum in (52) can be updated incrementally in constant time through the addition of $a_{\xi^{(i')}_{\psi_{i'}}} - a_{\xi^{(i')}_{(\psi_{i'}-1)}}$ (Lines 20–23 of Algorithm 5).

Suppose we find $\check{\eta} = \eta^{(i')}_{\psi_{i'}}$ for some i'. We then know that the minimum η^* is either equal to $\check{\eta}$ (Figure 11, left), or found within the quadratic segment immediately to its left (Figure 11, right).

11. For ease of exposition, we assume i' in (51) is unique, and deal with multiple choices of i' in Algorithm 5.

We thus decrement $\psi_{i'}$ (i.e., take one step back) so as to index the segment in question, set the right-hand side of (52) to zero, and solve for η' to obtain

$$\eta^* = \min\left(\check{\eta}, \frac{\lambda \boldsymbol{w}^\top \boldsymbol{p} + \frac{1}{n}\sum_{i=1}^n a_{\xi^{(i)}_{\psi_i}}}{-\lambda\|\boldsymbol{p}\|^2}\right). \tag{53}$$

This only takes constant time: we have cached $\boldsymbol{w}^\top \boldsymbol{p}$ and $\lambda\|\boldsymbol{p}\|^2$, and the sum in (53) can be obtained incrementally by adding $a_{\xi^{(i')}_{\psi_{i'}}} - a_{\xi^{(i')}_{(\psi_{i'}+1)}}$ to its last value in (52).

To locate $\check{\eta}$ we have to walk at most $O(n|\mathcal{Z}|)$ steps, each requiring $O(\log n)$ computation of argmin as in (51). Given $\check{\eta}$, the exact minimum η^* can be obtained in $O(1)$. Including the pre-processing cost of $O(n|\mathcal{Z}|\log|\mathcal{Z}|)$ (for invoking Algorithm 4), our exact multiclass line search therefore takes $O(n|\mathcal{Z}|(\log n|\mathcal{Z}|))$ time in the worst case. Algorithm 5 provides an implementation which instead of an index vector $\boldsymbol{\psi}$ directly uses the sorted stacks of subdifferentiable points and active lines produced by Algorithm 4. (The cost of reversing those stacks in Lines 6–8 of Algorithm 5 can easily be avoided through the use of double-ended queues.)

6.4 Multilabel Hinge Loss

Recently, there has been interest in extending the concept of the hinge loss to multilabel problems. Multilabel problems generalize the multiclass setting in that each training instance $\boldsymbol{x}_i$ is associated with a set of labels $\mathcal{Z}_i \subseteq \mathcal{Z}$ (Crammer and Singer, 2003b). For a uniform margin of separation τ, a hinge loss can be defined in this setting as follows:

$$l(\boldsymbol{x}_i, \mathcal{Z}_i, \boldsymbol{w}) := \max[0,\ \tau + \max_{z' \notin \mathcal{Z}_i} f(\boldsymbol{w}, \boldsymbol{x}_i, z') - \min_{z \in \mathcal{Z}_i} f(\boldsymbol{w}, \boldsymbol{x}_i, z)]. \tag{54}$$

We can generalize this to a not necessarily uniform label loss $\Delta(z', z) \geq 0$ as follows:

$$l(\boldsymbol{x}_i, \mathcal{Z}_i, \boldsymbol{w}) := \max_{\substack{(z,z'):\, z \in \mathcal{Z}_i \\ z' \notin \mathcal{Z}_i \backslash \{z\}}} [\Delta(z', z) + f(\boldsymbol{w}, \boldsymbol{x}_i, z') - f(\boldsymbol{w}, \boldsymbol{x}_i, z)], \tag{55}$$

where as before we require that $\Delta(z, z) = 0\ \forall z \in \mathcal{Z}$ so that by explicitly allowing $z' = z$ we can ensure that (55) remains non-negative. For a uniform margin $\Delta(z', z) = \tau\ \ \forall z' \neq z$ our multilabel hinge loss (55) reduces to the decoupled version (54), which in turn reduces to the multiclass hinge loss (42) if $\mathcal{Z}_i := \{z_i\}$ for all i.

For a given $\boldsymbol{w}$, let

$$\mathcal{Z}_i^* := \operatorname*{argmax}_{\substack{(z,z'):\, z \in \mathcal{Z}_i \\ z' \notin \mathcal{Z}_i \backslash \{z\}}} [\Delta(z', z) + f(\boldsymbol{w}, \boldsymbol{x}_i, z') - f(\boldsymbol{w}, \boldsymbol{x}_i, z)]$$

be the set of worst label pairs (possibly more than one) for the i^{th} training instance. The subdifferential of the multilabel analogue of L_2-regularized multiclass objective (43) can then be written just as in (44), with coefficients

$$\beta_{i,z} := \sum_{z':\, (z',z) \in \mathcal{Z}_i^*} \gamma^{(i)}_{z',z} - \sum_{z':\, (z,z') \in \mathcal{Z}_i^*} \gamma^{(i)}_{z,z'}, \quad \text{where} \quad (\forall i) \sum_{(z,z') \in \mathcal{Z}_i^*} \gamma^{(i)}_{z,z'} = 1 \ \text{ and } \ \gamma^{(i)}_{z,z'} \geq 0. \tag{56}$$

Now let $(z_i, z_i') := \text{argmax}_{(z,z') \in \mathcal{Z}_i^*} [\phi(\boldsymbol{x}_i, z') - \phi(\boldsymbol{x}_i, z)]^\top \boldsymbol{p}$ be a single steepest worst label pair in direction $\boldsymbol{p}$. We obtain $\arg\sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w})} \boldsymbol{g}^\top \boldsymbol{p}$ for our direction-finding procedure by picking, from the choices offered by (56), $\gamma_{z,z'}^{(i)} := \delta_{z,z_i} \delta_{z',z_i'}$.

Finally, the line search we described in Section 6.3 for the multiclass hinge loss can be extended in a straightforward manner to our multilabel setting. The only caveat is that now $\varrho_i(\eta) := l(\boldsymbol{x}_i, \mathcal{Z}_i, \boldsymbol{w} + \eta \boldsymbol{p})$ must be written as

$$\varrho_i(\eta) := \max_{\substack{(z,z'):\, z \in \mathcal{Z}_i \\ z' \notin \mathcal{Z}_i \setminus \{z\}}} [\underbrace{\Delta(z', z) + f(\boldsymbol{w}, \boldsymbol{x}_i, z') - f(\boldsymbol{w}, \boldsymbol{x}_i, z)}_{=:\, b_{z,z'}^{(i)}} + \eta \underbrace{(f(\boldsymbol{p}, \boldsymbol{x}_i, z') - f(\boldsymbol{p}, \boldsymbol{x}_i, z))}_{=:\, a_{z,z'}^{(i)}}]\,. \tag{57}$$

In the worst case, (57) could be the piecewise maximum of $O(|\mathcal{Z}|^2)$ lines, thus increasing the overall complexity of the line search. In practice, however, the set of true labels $\mathcal{Z}_i$ is usually small, typically of size 2 or 3 (cf. Crammer and Singer, 2003b, Figure 3). As long as $\forall i : |\mathcal{Z}_i| = O(1)$, our complexity estimates of Section 6.3.1 still apply.

7. Related Work

We discuss related work in two areas: nonsmooth convex optimization, and the problem of segmenting the pointwise maximum of a set of one-dimensional linear functions.

7.1 Nonsmooth Convex Optimization

There are four main approaches to nonsmooth convex optimization: quasi-Newton methods, bundle methods, stochastic dual methods, and smooth approximation. We discuss each of these briefly, and compare and contrast our work with the state of the art.

7.1.1 NONSMOOTH QUASI-NEWTON METHODS

These methods try to find a descent quasi-Newton direction at every iteration, and invoke a line search to minimize the one-dimensional convex function along that direction. We note that the line search routines we describe in Sections 4–6 are applicable to all such methods. An example of this class of algorithms is the work of Lukšan and Vlček (1999), who propose an extension of BFGS to nonsmooth convex problems. Their algorithm samples subgradients around non-differentiable points in order to obtain a descent direction. In many machine learning problems evaluating the objective function and its (sub)gradient is very expensive, making such an approach inefficient. In contrast, given a current iterate $\boldsymbol{w}_t$, our direction-finding routine (Algorithm 2) samples subgradients from the set $\partial J(\boldsymbol{w}_t)$ via the oracle. Since this avoids the cost of explicitly evaluating new (sub)gradients, it is computationally more efficient.

Recently, Andrew and Gao (2007) introduced a variant of LBFGS, the Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) algorithm, suitable for optimizing L_1-regularized log-linear models:

$$J(\boldsymbol{w}) := \lambda \|\boldsymbol{w}\|_1 + \underbrace{\frac{1}{n} \sum_{i=1}^{n} \ln(1 + e^{-z_i \boldsymbol{w}^\top \boldsymbol{x}_i})}_{\text{logistic loss}}, \tag{58}$$

where the logistic loss is smooth, but the regularizer is only subdifferentiable at points where $\boldsymbol{w}$ has zero elements. From the optimization viewpoint this objective is very similar to L_2-regularized hinge loss; the direction finding and line search methods that we discussed in Sections 3.2 and 3.3, respectively, can be applied to this problem with slight modifications.

OWL-QN is based on the observation that the L_1 regularizer is linear within any given orthant. Therefore, it maintains an approximation $\boldsymbol{B}^{\text{ow}}$ to the inverse Hessian of the logistic loss, and uses an efficient scheme to select orthants for optimization. In fact, its success greatly depends on its direction-finding subroutine, which demands a specially chosen subgradient $\boldsymbol{g}^{\text{ow}}$ (Andrew and Gao, 2007, Equation 4) to produce the quasi-Newton direction, $\boldsymbol{p}^{\text{ow}} = \pi(\boldsymbol{p}, \boldsymbol{g}^{\text{ow}})$, where $\boldsymbol{p} := -\boldsymbol{B}^{\text{ow}}\boldsymbol{g}^{\text{ow}}$ and the projection π returns a search direction by setting the i^{th} element of $\boldsymbol{p}$ to zero whenever $p_i g_i^{\text{ow}} > 0$. As shown in Section 8.4, the direction-finding subroutine of OWL-QN can be replaced by our Algorithm 2, which makes OWL-QN more robust to the choice of subgradients.

7.1.2 BUNDLE METHODS

Bundle method solvers (Hiriart-Urruty and Lemaréchal, 1993) use past (sub)gradients to build a model of the objective function. The (sub)gradients are used to lower-bound the objective by a piecewise linear function which is minimized to obtain the next iterate. This fundamentally differs from the BFGS approach of using past gradients to approximate the (inverse) Hessian, hence building a quadratic model of the objective function.

Bundle methods have recently been adapted to the machine learning context, where they are known as SVMStruct (Tsochantaridis et al., 2005) *resp.* BMRM (Smola et al., 2007). One notable feature of these variants is that they do not employ a line search. This is justified by noting that a line search involves computing the value of the objective function multiple times, a potentially expensive operation in machine learning applications.

Franc and Sonnenburg (2008) speed up the convergence of SVMStruct for L_2-regularized binary hinge loss. The main idea of their optimized cutting plane algorithm, OCAS, is to perform a line search along the line connecting two successive iterates of a bundle method solver. Recently they have extended OCAS to multiclass classification (Franc and Sonnenburg, 2009). Although developed independently, their line search methods for both settings are very similar to the methods we describe in Sections 4.2.1 and 6.3.1, respectively. In particular, their line search for multiclass classification also involves segmenting the pointwise maximum of r 1-D linear functions (cf. Section 5), though the $O(r^2)$ time complexity of their method is worse than our $O(r \log r)$.

7.1.3 STOCHASTIC DUAL METHODS

Distinct from the above two classes of primal algorithms are methods which work in the dual domain. A prominent member of this class is the LaRank algorithm of Bordes et al. (2007), which achieves state-of-the-art results on multiclass classification problems. While dual algorithms are very competitive on clean data sets, they tend to be slow when given noisy data.

7.1.4 SMOOTH APPROXIMATION

Another possible way to bypass the complications caused by the nonsmoothness of an objective function is to work on a smooth approximation instead—see for instance the recent work of Nesterov (2005) and Nemirovski (2005). Some machine learning applications have also been pursued along these lines (Lee and Mangasarian, 2001; Zhang and Oles, 2001). Although this approach can be effective, it is unclear how to build a smooth approximation in general. Furthermore, smooth approximations often sacrifice dual sparsity, which often leads to better generalization performance on the test data, and also may be needed to prove generalization bounds.

7.2 Segmenting the Pointwise Maximum of 1-D Linear Functions

The problem of computing the line segments that comprise the pointwise maximum of a given set of line segments has received attention in the area of computational geometry; see Agarwal and Sharir (2000) for a survey. Hershberger (1989) for instance proposed a divide-and-conquer algorithm for this problem with the same time complexity as our Algorithm 4. The Hershberger (1989) algorithm solves a slightly harder problem—his function is the pointwise maximum of line segments, as opposed to our lines—but our algorithm is conceptually simpler and easier to implement.

A similar problem has also been studied under the banner of kinetic data structures by Basch (1999), who proposed a heap-based algorithm for this problem and proved a worst-case $O(r \log^2 r)$ bound, where r is the number of line segments. Basch (1999) also claims that the lower bound is $O(r \log r)$; our Algorithm 4 achieves this bound.

8. Experiments

We evaluated the performance of our subLBFGS algorithm with, and compared it to other state-of-the-art nonsmooth optimization methods on L_2-regularized binary, multiclass, and multilabel hinge loss minimization problems. We also compared OWL-QN with a variant that uses our direction-finding routine on L_1-regularized logistic loss minimization tasks. On strictly convex problems such as these every convergent optimizer will reach the same solution; comparing generalisation performance is therefore pointless. Hence we concentrate on empirically evaluating the convergence behavior (objective function value *vs.* CPU seconds). All experiments were carried out on a Linux machine with dual 2.4 GHz Intel Core 2 processors and 4 GB of RAM.

In all experiments the regularization parameter was chosen from the set $10^{\{-6,-5,\cdots,-1\}}$ so as to achieve the highest prediction accuracy on the test data set, while convergence behavior (objective function value *vs.* CPU seconds) is reported on the training data set. To see the influence of the regularization parameter λ, we also compared the time required by each algorithm to reduce the objective function value to within 2% of the optimal value.[12] For all algorithms the initial iterate $\boldsymbol{w}_0$ was set to $\mathbf{0}$. Open source C++ code implementing our algorithms and experiments is available for download from `http://www.cs.adelaide.edu.au/~jinyu/Code/nonsmoothOpt.tar.gz`.

The subgradient for the construction of the subLBFGS search direction (cf. Line 12 of Algorithm 1) was chosen arbitrarily from the subdifferential. For the binary hinge loss minimization (Section 8.3), for instance, we picked an arbitrary subgradient by randomly setting the coefficient $\beta_i \; \forall i \in \mathcal{M}$ in (32) to either 0 or 1.

8.1 Convergence Tolerance of the Direction-Finding Procedure

The convergence tolerance ϵ of Algorithm 2 controls the precision of the solution to the direction-finding problem (11): lower tolerance may yield a better search direction. Figure 13 (left) shows

12. For L_1-regularized logistic loss minimization, the "optimal" value was the final objective function value achieved by the OWL-QN* algorithm (cf. Section 8.4). In all other experiments, it was found by running subLBFGS for 10^4 seconds, or until its relative improvement over 5 iterations was less than 10^{-8}.

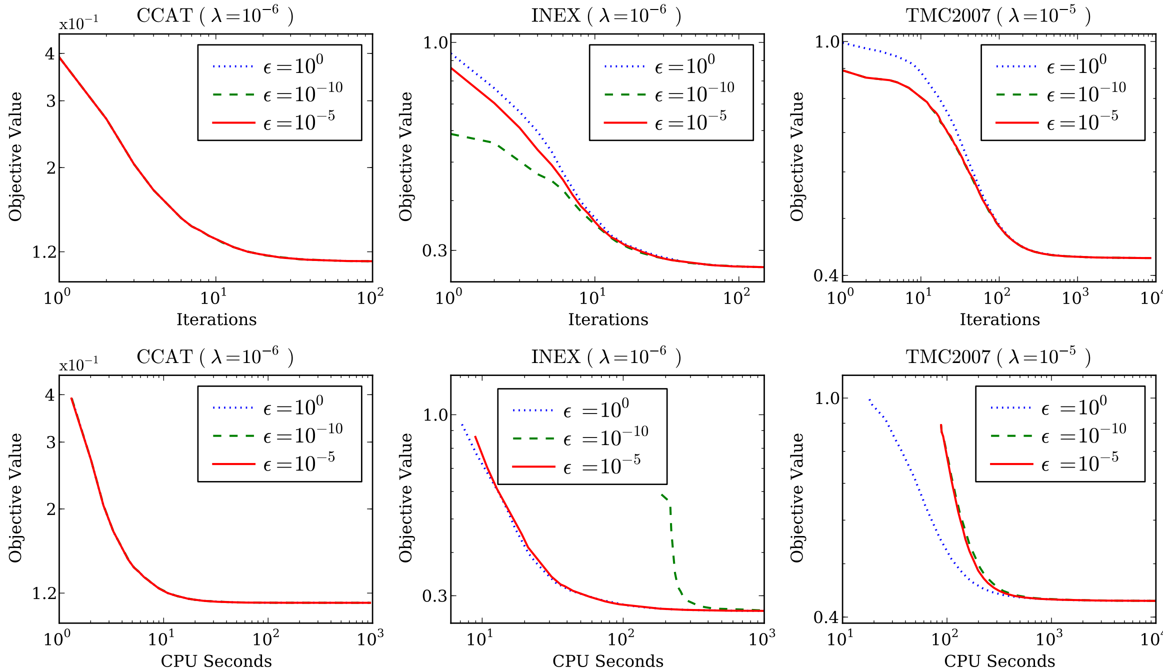


Figure 13: Performance of subLBFGS with varying direction-finding tolerance ϵ in terms of objective function value *vs.* number of iterations (top row) *resp.* CPU seconds (bottom row) on sample L_2-regularized risk minimization problems with binary (left), multiclass (center), and multilabel (right) hinge losses.

that on binary classification problems, subLBFGS is not sensitive to the choice of ϵ (i.e., the quality of the search direction). This is due to the fact that $\partial J(\boldsymbol{w})$ as defined in (32) is usually dominated by its constant component $\bar{\boldsymbol{w}}$; search directions that correspond to different choices of ϵ therefore can not differ too much from each other. In the case of multiclass and multilabel classification, where the structure of $\partial J(\boldsymbol{w})$ is more complicated, we can see from Figure 13 (top center and right) that a better search direction can lead to faster convergence in terms of iteration numbers. However, this is achieved at the cost of more CPU time spent in the direction-finding routine. As shown in Figure 13 (bottom center and right), extensively optimizing the search direction actually slows down convergence in terms of CPU seconds. We therefore used an intermediate value of $\epsilon = 10^{-5}$ for all our experiments, except that for multiclass and multilabel classification problems we relaxed the tolerance to 1.0 at the initial iterate $\boldsymbol{w} = \mathbf{0}$, where the direction-finding oracle $\arg\sup_{\boldsymbol{g}\in\partial J(\mathbf{0})} \boldsymbol{g}^\top \boldsymbol{p}$ is expensive to compute, due to the large number of extreme points in $\partial J(\mathbf{0})$.

8.2 Size of SubLBFGS Buffer

The size m of the subLBFGS buffer determines the number of parameter and gradient displacement vectors $\boldsymbol{s}_t$ and $\boldsymbol{y}_t$ used in the construction of the quasi-Newton direction. Figure 14 shows that the performance of subLBFGS is not sensitive to the particular value of m within the range $5 \leq m \leq 25$.

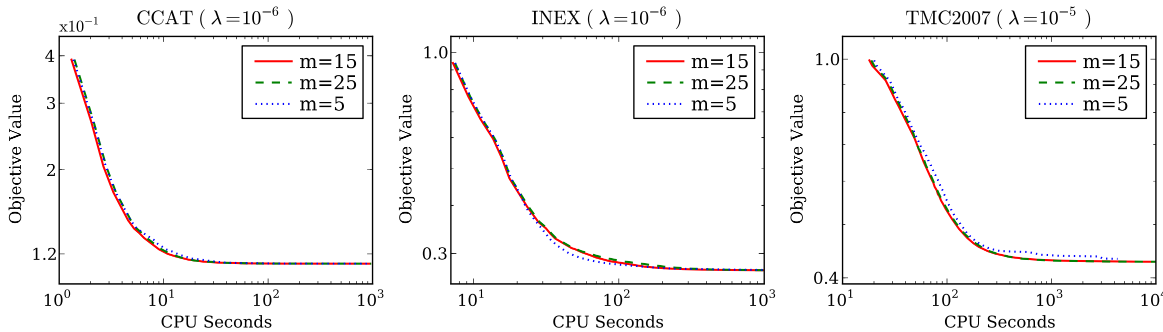


Figure 14: Performance of subLBFGS with varying buffer size on sample L_2-regularized risk minimization problems with binary (left), multiclass (center), and multilabel hinge losses (right).

Data Set	Train/Test Set Size	Dimensionality	Sparsity
Covertype	522911/58101	54	77.8%
CCAT	781265/23149	47236	99.8%
Astro-physics	29882/32487	99757	99.9%
MNIST-binary	60000/10000	780	80.8%
Adult9	32561/16281	123	88.7%
Real-sim	57763/14438	20958	99.8%
Leukemia	38/34	7129	00.0%

Table 1: The binary data sets used in our experiments of Sections 2, 8.3, and 8.4.

We therefore simply set $m = 15$ *a priori* for all subsequent experiments; this is a typical value for LBFGS (Nocedal and Wright, 1999).

8.3 L_2-Regularized Binary Hinge Loss

For our first set of experiments, we applied subLBFGS with exact line search (Algorithm 3) to the task of L_2-regularized binary hinge loss minimization. Our control methods are the bundle method solver BMRM (Teo et al., 2010) and the optimized cutting plane algorithm OCAS (Franc and Sonnenburg, 2008),[13] both of which were shown to perform competitively on this task. SVMStruct (Tsochantaridis et al., 2005) is another well-known bundle method solver that is widely used in the machine learning community. For L_2-regularized optimization problems BMRM is identical to SVMStruct, hence we omit comparisons with SVMStruct.

Table 1 lists the six data sets we used: The Covertype data set of Blackard, Jock & Dean,[14] CCAT from the Reuters RCV1 collection,[15] the Astro-physics data set of abstracts of scientific papers from the Physics ArXiv (Joachims, 2006), the MNIST data set of handwritten digits[16] with

13. The source code of OCAS (version 0.6.0) was obtained from http://www.shogun-toolbox.org.
14. Data set can be found at http://kdd.ics.uci.edu/databases/covertype/covertype.html.
15. Data set can be found at http://www.daviddlewis.com/resources/testcollections/rcv1.
16. Data set can be found at http://yann.lecun.com/exdb/mnist.

Data Set	L_1-reg. logistic loss λ_{L_1}	k_{L_1}	$k_{L_1 r}$	L_2-reg. binary loss λ_{L_2}	k_{L_2}
Covertype	10^{-5}	1	2	10^{-6}	0
CCAT	10^{-6}	284	406	10^{-6}	0
Astro-physics	10^{-5}	1702	1902	10^{-4}	0
MNIST-binary	10^{-4}	55	77	10^{-6}	0
Adult9	10^{-4}	2	6	10^{-5}	1
Real-sim	10^{-6}	1017	1274	10^{-5}	1

Table 2: Regularization parameter λ and overall number k of direction-finding iterations in our experiments of Sections 8.3 and 8.4, respectively.

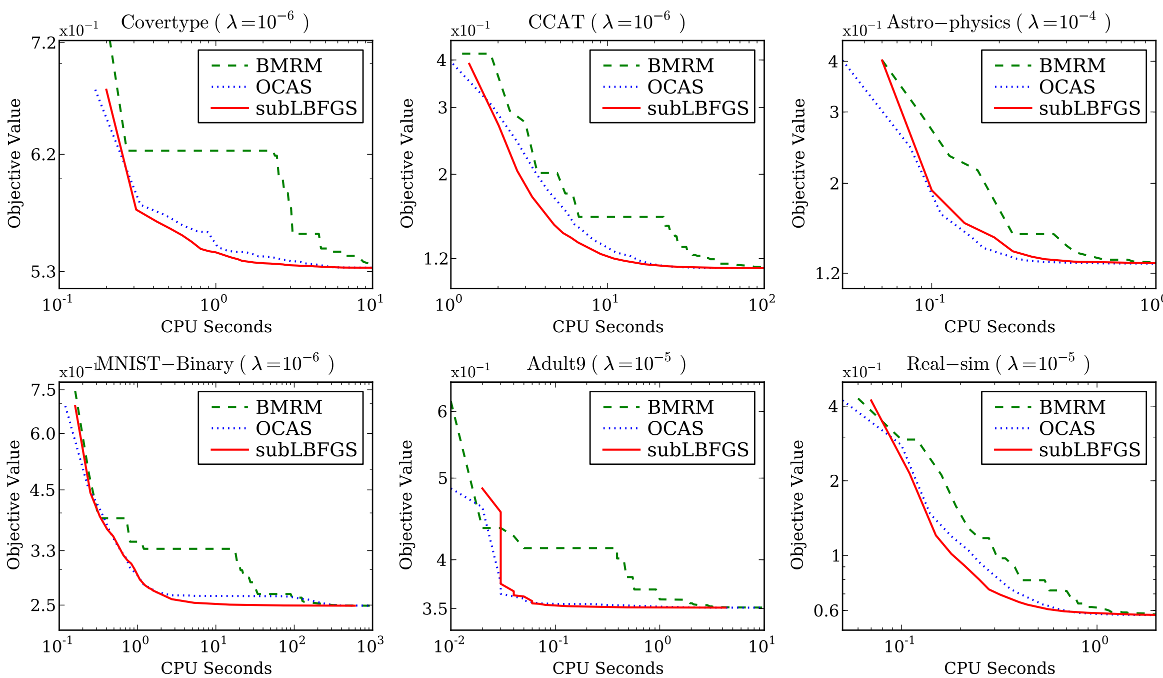


Figure 15: Objective function value *vs.* CPU seconds on L_2-regularized binary hinge loss minimization tasks.

two classes: even and odd digits, the Adult9 data set of census income data,[17] and the Real-sim data set of real *vs.* simulated data.[17] Table 2 lists our parameter settings, and reports the overall number k_{L_2} of iterations through the direction-finding loop (Lines 6–13 of Algorithm 2) for each data set. The very small values of k_{L_2} indicate that on these problems subLBFGS only rarely needs to correct its initial guess of a descent direction.

It can be seen from Figure 15 that subLBFGS (solid) reduces the value of the objective considerably faster than BMRM (dashed). On the binary MNIST data set, for instance, the objective

17. Data set can be found at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`.

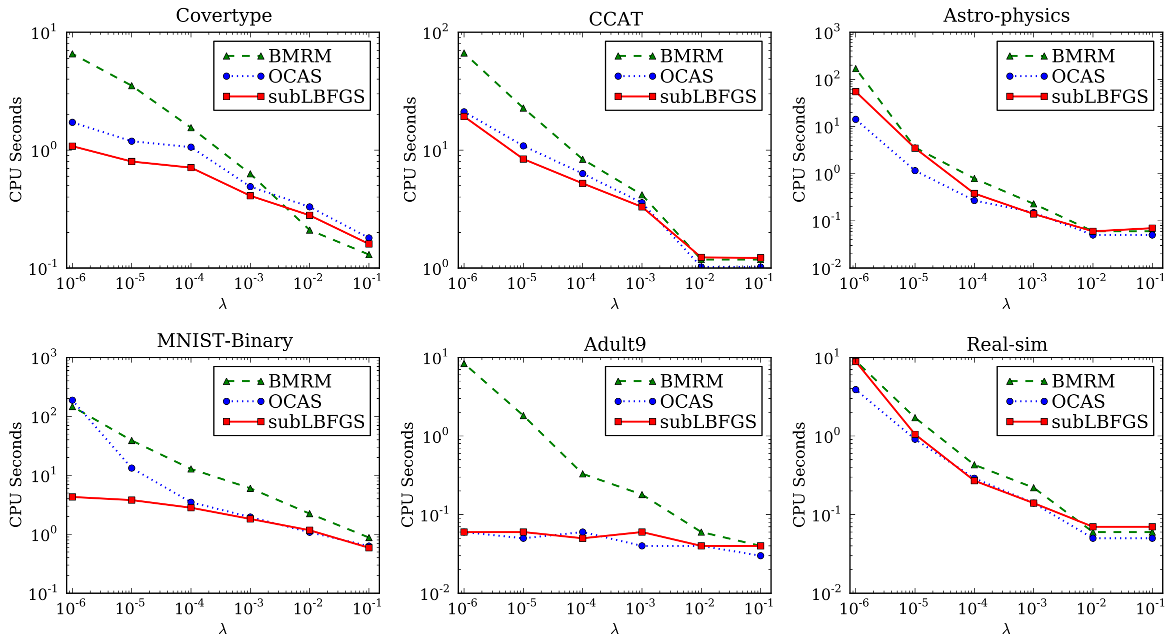


Figure 16: Regularization parameter $\lambda \in \{10^{-6}, \cdots, 10^{-1}\}$ *vs.* CPU seconds taken to reduce the objective function to within 2% of the optimal value on L_2-regularized binary hinge loss minimization tasks.

function value of subLBFGS after 10 CPU seconds is 25% lower than that of BMRM. In this set of experiments the performance of subLBFGS and OCAS (dotted) is very similar.

Figure 16 shows that all algorithms generally converge faster for larger values of the regularization constant λ. However, in most cases subLBFGS converges faster than BMRM across a wide range of λ values, exhibiting a speedup of up to more than two orders of magnitude. SubLBFGS and OCAS show similar performance here: for small values of λ, OCAS converges slightly faster than subLBFGS on the Astro-physics and Real-sim data sets but is outperformed by subLBFGS on the Covertype, CCAT, and binary MNIST data sets.

8.4 L_1-Regularized Logistic Loss

To demonstrate the utility of our direction-finding routine (Algorithm 2) in its own right, we plugged it into the OWL-QN algorithm (Andrew and Gao, 2007)[18] as an alternative direction-finding method such that $\boldsymbol{p}^{\text{ow}} = \texttt{descentDirection}(\boldsymbol{g}^{\text{ow}}, \epsilon, k_{\max})$, and compared this variant (denoted OWL-QN*) with the original (cf. Section 7.1) on L_1-regularized minimization of the logistic loss (58), on the same data sets as in Section 8.3.

An oracle that supplies $\arg\sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top \boldsymbol{p}$ for this objective is easily constructed by noting that (58) is nonsmooth whenever at least one component of the parameter vector $\boldsymbol{w}$ is zero. Let $w_i = 0$ be such a component; the corresponding component of the subdifferential $\partial\, \lambda\|\boldsymbol{w}\|_1$ of the L_1

18. The source code of OWL-QN (original release) was obtained from Microsoft Research through `http://tinyurl.com/p774cx`.

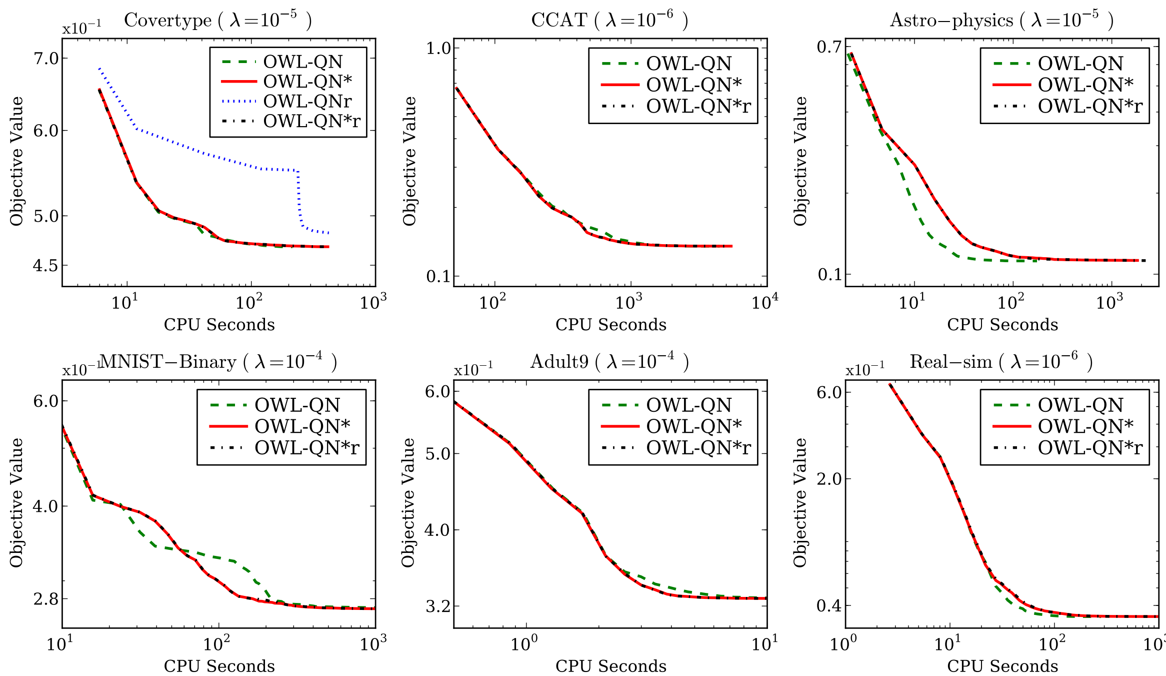


Figure 17: Objective function value *vs.* CPU seconds on L_1-regularized logistic loss minimization tasks.

regularizer is the interval $[-\lambda, \lambda]$. The supremum of $\boldsymbol{g}^\top \boldsymbol{p}$ is attained at the interval boundary whose sign matches that of the corresponding component of the direction vector $\boldsymbol{p}$, that is, at $\lambda \operatorname{sign}(p_i)$.

Using the stopping criterion suggested by Andrew and Gao (2007), we ran experiments until the averaged relative change in objective function value over the previous 5 iterations fell below 10^{-5}. As shown in Figure 17, the only clear difference in convergence between the two algorithms is found on the Astro-physics data set where OWL-QN* is outperformed by the original OWL-QN method. This is because finding a descent direction via Algorithm 2 is particularly difficult on the Astro-physics data set (as indicated by the large inner loop iteration number k_{L_1} in Table 2); the slowdown on this data set can also be found in Figure 18 for other values of λ. Although finding a descent direction can be challenging for the generic direction-finding routine of OWL-QN*, in the following experiment we show that this routine is very robust to the choice of initial subgradients.

To examine the algorithms' sensitivity to the choice of subgradients, we also ran them with subgradients randomly chosen from the set $\partial J(\boldsymbol{w})$ (as opposed to the specially chosen subgradient $\boldsymbol{g}^{\text{ow}}$ used in the previous set of experiments) fed to their corresponding direction-finding routines. OWL-QN relies heavily on its particular choice of subgradients, hence breaks down completely under these conditions: the only data set where we could even plot its (poor) performance was Covertype (dotted "OWL-QNr" line in Figure 17). Our direction-finding routine, by contrast, is self-correcting and thus not affected by this manipulation: the curves for OWL-QN*r lie on top of those for OWL-QN*. Table 2 shows that in this case more direction-finding iterations are needed though: $k_{L_1 r} > k_{L_1}$. This empirically confirms that as long as $\arg\sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w})} \boldsymbol{g}^\top \boldsymbol{p}$ is given, Algorithm 2

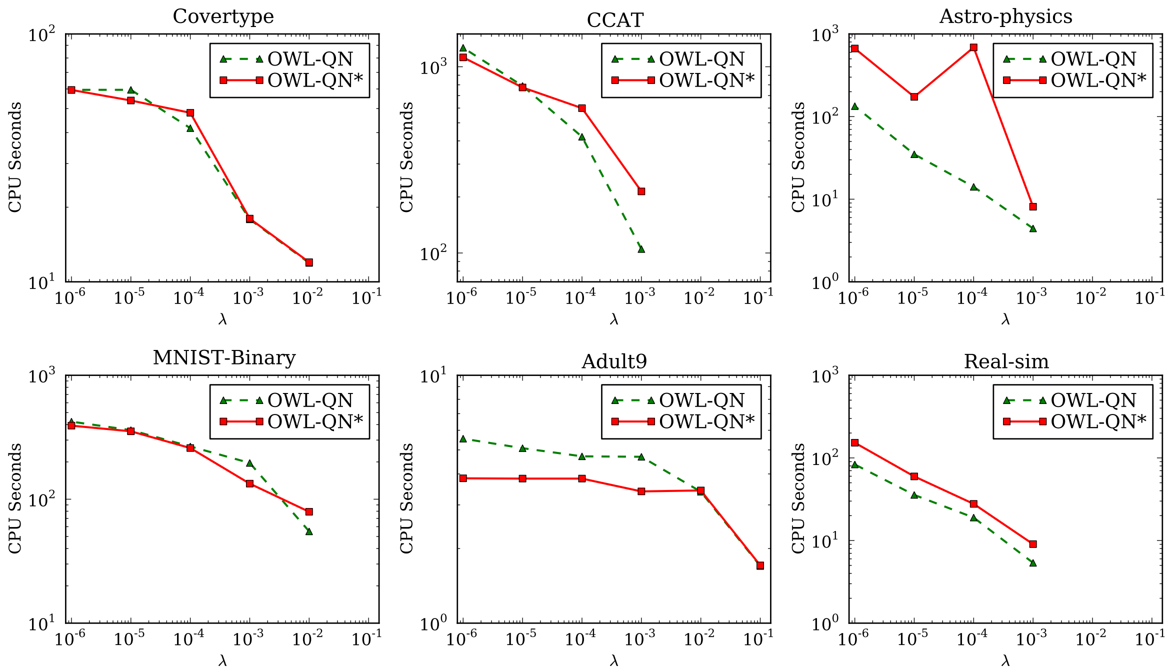


Figure 18: Regularization parameter $\lambda \in \{10^{-6}, \cdots, 10^{-1}\}$ *vs.* CPU seconds taken to reduce the objective function to within 2% of the optimal value on L_1-regularized logistic loss minimization tasks. (No point is plotted if the initial parameter $\boldsymbol{w}_0 = \mathbf{0}$ is already optimal.)

can indeed be used as a generic quasi-Newton direction-finding routine that is able to recover from a poor initial choice of subgradients.

8.5 L_2-Regularized Multiclass and Multilabel Hinge Loss

We incorporated our exact line search of Section 6.3.1 into both subLBFGS and OCAS (Franc and Sonnenburg, 2008), thus enabling them to deal with multiclass and multilabel losses. We refer to our generalized version of OCAS as line search BMRM (ls-BMRM). Using the variant of the multiclass and multilabel hinge loss which enforces a uniform margin of separation ($\Delta(z, z') = 1$ $\forall z \neq z'$), we experimentally evaluated both algorithms on a number of publicly available data sets (Table 3). All multiclass data sets except INEX were downloaded from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html`, while the multilabel data sets hail from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html`. INEX (Maes et al., 2007) is available from `http://webia.lip6.fr/~bordes/mywiki/doku.php?id=multiclass_data`. The original RCV1 data set consists of 23149 training instances, of which we used 21149 instances for training and the remaining 2000 for testing.

8.5.1 PERFORMANCE ON MULTICLASS PROBLEMS

This set of experiments is designed to demonstrate the convergence properties of multiclass subLBFGS, compared to the BMRM bundle method (Teo et al., 2010) and ls-BMRM. Figure 19 shows

Data Set	Train/Test Set Size	Dimensionality	$\lvert\mathcal{Z}\rvert$	Sparsity	λ	k
Letter	16000/4000	16	26	0.0%	10^{-6}	65
USPS	7291/2007	256	10	3.3%	10^{-3}	14
Protein	14895/6621	357	3	70.7%	10^{-2}	1
MNIST	60000/10000	780	10	80.8%	10^{-3}	1
INEX	6053/6054	167295	18	99.5%	10^{-6}	5
News20	15935/3993	62061	20	99.9%	10^{-2}	12
Scene	1211/1196	294	6	0.0%	10^{-1}	14
TMC2007	21519/7077	30438	22	99.7%	10^{-5}	19
RCV1	21149/2000	47236	103	99.8%	10^{-5}	4

Table 3: The multiclass (top 6 rows) and multilabel (bottom 3 rows) data sets used, values of the regularization parameter, and overall number k of direction-finding iterations in our experiments of Section 8.5.

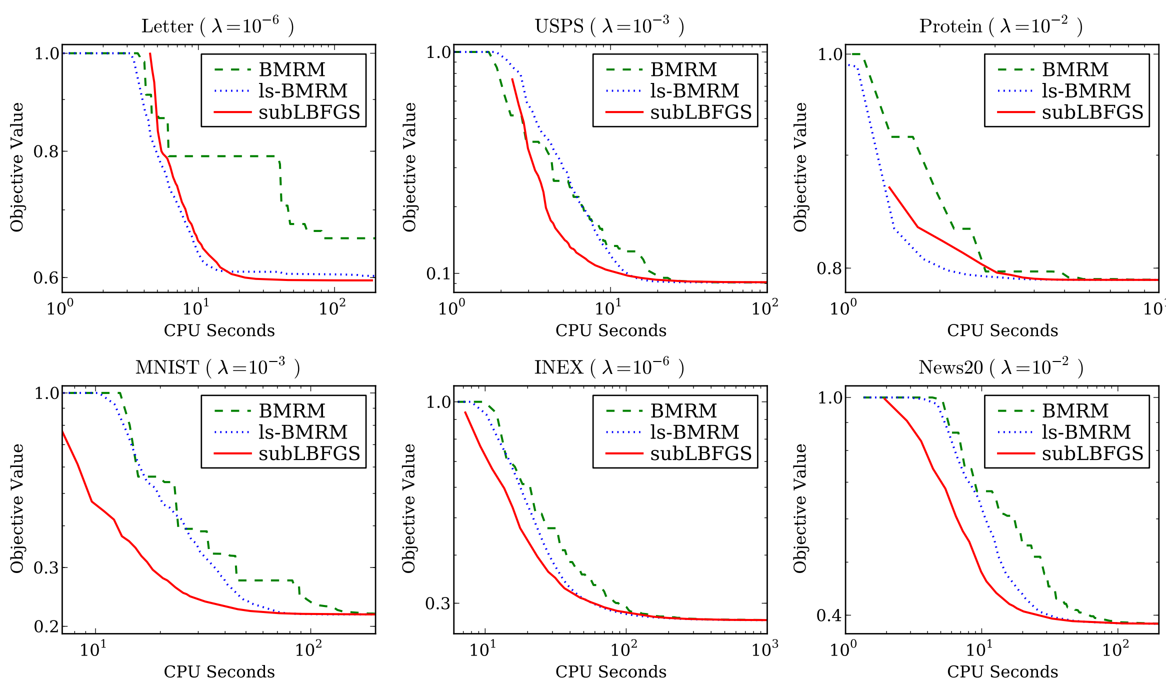


Figure 19: Objective function value *vs.* CPU seconds on L_2-regularized multiclass hinge loss minimization tasks.

that subLBFGS outperforms BMRM on all data sets. On 4 out of 6 data sets, subLBFGS outperforms ls-BMRM as well early on but slows down later, for an overall performance comparable to ls-BMRM. On the MNIST data set, for instance, subLBFGS takes only about half as much CPU time as ls-BMRM to reduce the objective function value to 0.3 (about 50% above the optimal value),

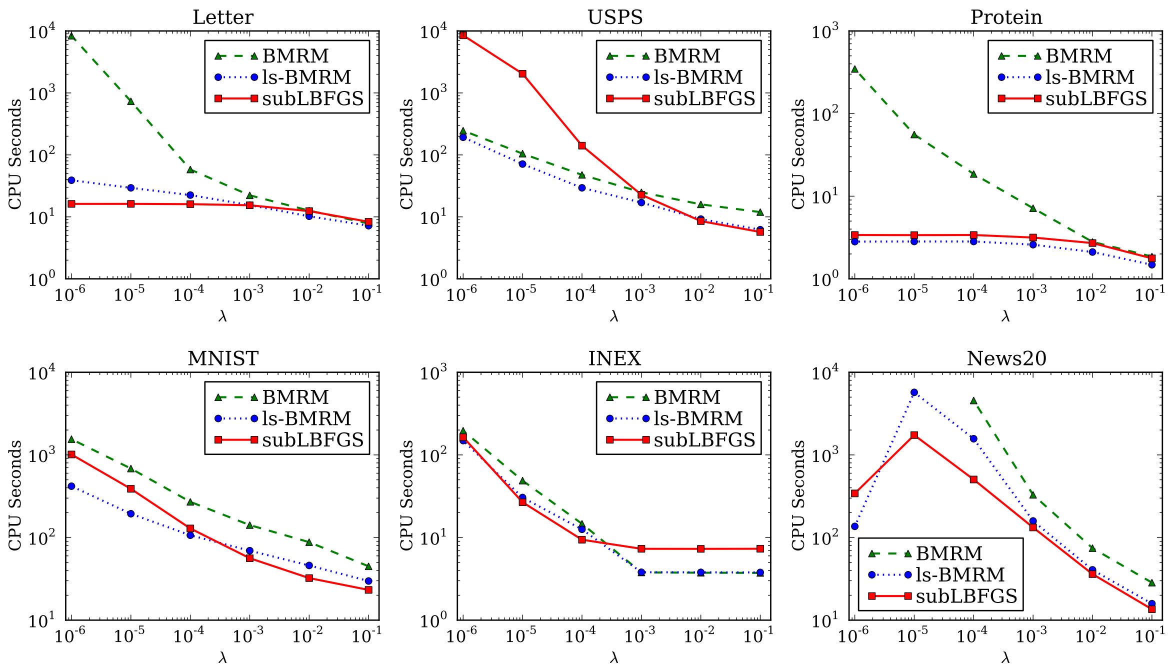


Figure 20: Regularization parameter $\lambda \in \{10^{-6}, \cdots, 10^{-1}\}$ *vs.* CPU seconds taken to reduce the objective function to within 2% of the optimal value. (No point is plotted if an algorithm failed to reach the threshold value within 10^4 seconds.)

yet both algorithms reach within 2% of the optimal value at about the same time (Figure 20, bottom left). We hypothesize that subLBFGS' local model (10) of the objective function facilitates rapid early improvement but is less appropriate for final convergence to the optimum (cf. the discussion in Section 9). Bundle methods, on the other hand, are slower initially because they need to accumulate a sufficient number of gradients to build a faithful piecewise linear model of the objective function. These results suggest that a hybrid approach that first runs subLBFGS then switches to ls-BMRM may be promising.

Similar to what we saw in the binary setting (Figure 16), Figure 20 shows that all algorithms tend to converge faster for large values of λ. Generally, subLBFGS converges faster than BMRM across a wide range of λ values; for small values of λ it can greatly outperform BMRM (as seen on Letter, Protein, and News20). The performance of subLBFGS is worse than that of BMRM in two instances: on USPS for small values of λ, and on INEX for large values of λ. The poor performance on USPS may be caused by a limitation of subLBFGS' local model (10) that causes it to slow down on final convergence. On the INEX data set, the initial point $\boldsymbol{w}_0 = \mathbf{0}$ is nearly optimal for large values of λ; in this situation there is no advantage in using subLBFGS.

Leveraging its exact line search (Algorithm 5), ls-BMRM is competitive on all data sets and across all λ values, exhibiting performance comparable to subLBFGS in many cases. From Figure 20 we find that BMRM never outperforms both subLBFGS and ls-BMRM.

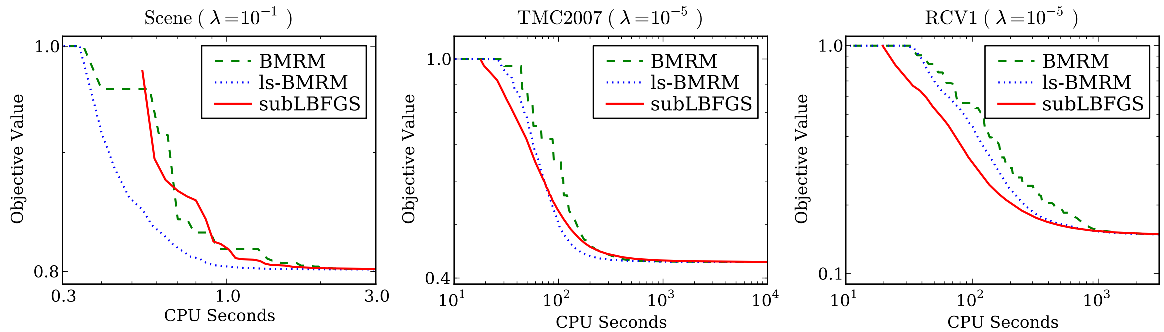


Figure 21: Objective function value *vs.* CPU seconds in L_2-regularized multilabel hinge loss minimization tasks.

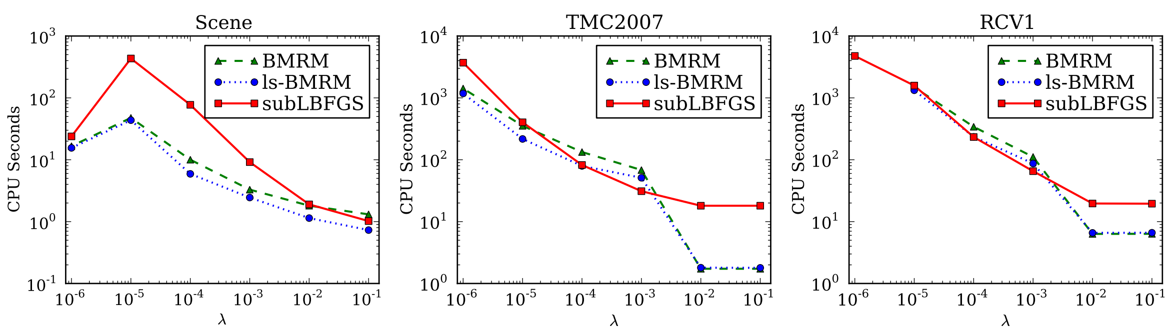


Figure 22: Regularization parameter $\lambda \in \{10^{-6}, \cdots, 10^{-1}\}$ *vs.* CPU seconds taken to reduce the objective function to within 2% of the optimal value. (No point is plotted if an algorithm failed to reach the threshold value within 10^4 seconds.)

8.5.2 PERFORMANCE ON MULTILABEL PROBLEMS

For our final set of experiments we turn to the multilabel setting. Figure 21 shows that on the Scene data set the performance of subLBFGS is similar to that of BMRM, while on the larger TMC2007 and RCV1 sets, subLBFGS outperforms both of its competitors initially but slows down later on, resulting in performance no better than BMRM. Comparing performance across different values of λ (Figure 22), we find that in many cases subLBFGS requires more time than its competitors to reach within 2% of the optimal value, and in contrast to the multiclass setting, here ls-BMRM only performs marginally better than BMRM. The primary reason for this is that the exact line search used by ls-BMRM and subLBFGS requires substantially more computational effort in the multilabel than in the multiclass setting. There is an inherent trade-off here: subLBFGS and ls-BMRM expend computation in an exact line search, while BMRM focuses on improving its local model of the objective function instead. In situations where the line search is very expensive, the latter strategy seems to pay off.

9. Discussion and Outlook

We proposed subBFGS (resp., subLBFGS), an extension of the BFGS quasi-Newton method (resp., its limited-memory variant), for handling nonsmooth convex optimization problems, and proved its global convergence in objective function value. We applied our algorithm to a variety of machine learning problems employing the L_2-regularized binary hinge loss and its multiclass and multilabel generalizations, as well as L_1-regularized risk minimization with logistic loss. Our experiments show that our algorithm is versatile, applicable to many problems, and often outperforms specialized solvers.

Our solver is easy to parallelize: The master node computes the search direction and transmits it to the slaves. The slaves compute the (sub)gradient and loss value on subsets of data, which is aggregated at the master node. This information is used to compute the next search direction, and the process repeats. Similarly, the line search, which is the expensive part of the computation on multiclass and multilabel problems, is easy to parallelize: The slaves run Algorithm 4 on subsets of the data; the results are fed back to the master which can then run Algorithm 5 to compute the step size.

In many of our experiments we observe that subLBFGS decreases the objective function rapidly at the beginning but slows down closer to the optimum. We hypothesize that this is due to an averaging effect: Initially (i.e., when sampled sparsely at a coarse scale) a superposition of many hinges looks sufficiently similar to a smooth function for optimization of a quadratic local model to work well (cf. Figure 6). Later on, when the objective is sampled at finer resolution near the optimum, the few nearest hinges begin to dominate the picture, making a smooth local model less appropriate.

Even though the local model (10) of sub(L)BFGS is nonsmooth, it only explicitly models the hinges at its present location—all others are subject to smooth quadratic approximation. Apparently this strategy works sufficiently well during early iterations to provide for rapid improvement on multiclass problems, which typically comprise a large number of hinges. The exact location of the optimum, however, may depend on individual nearby hinges which are not represented in (10), resulting in the observed slowdown.

Bundle method solvers, by contrast, exhibit slow initial progress but tend to be competitive asymptotically. This is because they build a piecewise linear lower bound of the objective function, which initially is not very good but through successive tightening eventually becomes a faithful model. To take advantage of this we are contemplating hybrid solvers that switch over from sub(L)BFGS to a bundle method as appropriate.

While bundle methods like BMRM have an exact, implementable stopping criterion based on the duality gap, no such stopping criterion exists for BFGS and other quasi-Newton algorithms. Therefore, it is customary to use the relative change in function value as an implementable stopping criterion. Developing a stopping criterion for sub(L)BFGS based on duality arguments remains an important open question.

sub(L)BFGS relies on an efficient exact line search. We proposed such line searches for the multiclass hinge loss and its extension to the multilabel setting, based on a conceptually simple yet optimal algorithm to segment the pointwise maximum of lines. A crucial assumption we had to make is that the number $|\mathcal{Z}|$ of labels is manageable, as it takes $O(|\mathcal{Z}| \log |\mathcal{Z}|)$ time to identify the hinges associated with each training instance. In certain structured prediction problems (Tsochantaridis et al., 2005) which have recently gained prominence in machine learning, the set $\mathcal{Z}$ could be exponentially large—for instance, predicting binary labels on a chain of length n produces 2^n possible labellings. Clearly our line searches are not efficient in such cases; we are investigating trust region variants of sub(L)BFGS to bridge this gap.

Finally, to put our contributions in perspective, recall that we modified three aspects of the standard BFGS algorithm, namely the quadratic model (Section 3.1), the descent direction finding (Section 3.2), and the Wolfe conditions (Section 3.3). Each of these modifications is versatile enough to be used as a component in other nonsmooth optimization algorithms. This not only offers the promise of improving existing algorithms, but may also help clarify connections between them. We hope that our research will focus attention on the core subroutines that need to be made more efficient in order to handle larger and larger data sets.

Acknowledgments

A short version of this paper was presented at the 2008 ICML conference (Yu et al., 2008). We thank Choon Hui Teo for many useful discussions and help with implementation issues, Xinhua Zhang for proofreading our manuscript, and the anonymous reviewers of both ICML and JMLR for their useful feedback which helped improve this paper. We thank John R. Birge for pointing us to his work (Birge et al., 1998) which led us to the convergence proof in Appendix D.

This publication only reflects the authors' views. All authors were with NICTA and the Australian National University for parts of their work on it. NICTA is funded by the Australian Government's Backing Australia's Ability and Centre of Excellence programs. This work was also supported in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886.

Appendix A. Bundle Search for a Descent Direction

Recall from Section 3.2 that at a subdifferential point $\boldsymbol{w}$ our goal is to find a descent direction $\boldsymbol{p}^*$ which minimizes the pseudo-quadratic model:[19]

$$M(\boldsymbol{p}) := \tfrac{1}{2}\boldsymbol{p}^\top \boldsymbol{B}^{-1}\boldsymbol{p} + \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top \boldsymbol{p}. \tag{59}$$

This is generally intractable due to the presence of a supremum over the entire subdifferential $\partial J(\boldsymbol{w})$. We therefore propose a bundle-based descent direction finding procedure (Algorithm 2) which progressively approaches $M(\boldsymbol{p})$ from below via a series of convex functions $M^{(1)}(\boldsymbol{p}), \cdots, M^{(i)}(\boldsymbol{p})$, each taking the same form as $M(\boldsymbol{p})$ but with the supremum defined over a countable subset of $\partial J(\boldsymbol{w})$. At iteration i our convex lower bound $M^{(i)}(\boldsymbol{p})$ takes the form

$$\begin{aligned} M^{(i)}(\boldsymbol{p}) &:= \tfrac{1}{2}\boldsymbol{p}^\top \boldsymbol{B}^{-1}\boldsymbol{p} + \sup_{\boldsymbol{g}\in\mathcal{V}^{(i)}} \boldsymbol{g}^\top \boldsymbol{p}, \text{ where} \\ \mathcal{V}^{(i)} &:= \{\boldsymbol{g}^{(j)} : j \le i,\ i,j\ \in \mathbb{N}\} \subseteq \partial J(\boldsymbol{w}). \end{aligned} \tag{60}$$

Given an iterate $\boldsymbol{p}^{(j-1)} \in \mathbb{R}^d$ we find a *violating subgradient* $\boldsymbol{g}^{(j)}$ via

$$\boldsymbol{g}^{(j)} := \operatorname*{arg\,sup}_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top \boldsymbol{p}^{(j-1)}. \tag{61}$$

19. For ease of exposition we are suppressing the iteration index t here.

Violating subgradients recover the true objective $M(\boldsymbol{p})$ at the iterates $\boldsymbol{p}^{(j-1)}$:

$$M(\boldsymbol{p}^{(j-1)}) = M^{(j)}(\boldsymbol{p}^{(j-1)}) = \tfrac{1}{2}\,\boldsymbol{p}^{(j-1)\top}\boldsymbol{B}^{-1}\boldsymbol{p}^{(j-1)} + \boldsymbol{g}^{(j)\top}\boldsymbol{p}^{(j-1)}. \tag{62}$$

To produce the iterates $\boldsymbol{p}^{(i)}$, we rewrite $\min_{\boldsymbol{p}\in\mathbb{R}^d} M^{(i)}(\boldsymbol{p})$ as a constrained optimization problem (19), which allows us to write the Lagrangian of (60) as

$$L^{(i)}(\boldsymbol{p},\xi,\boldsymbol{\alpha}) := \tfrac{1}{2}\,\boldsymbol{p}^\top\boldsymbol{B}^{-1}\boldsymbol{p} + \xi - \boldsymbol{\alpha}^\top(\xi\mathbf{1} - {\boldsymbol{G}^{(i)}}^\top\boldsymbol{p}), \tag{63}$$

where $\boldsymbol{G}^{(i)} := [\boldsymbol{g}^{(1)}, \boldsymbol{g}^{(2)}, \ldots, \boldsymbol{g}^{(i)}] \in \mathbb{R}^{d\times i}$ collects past violating subgradients, and $\boldsymbol{\alpha}$ is a column vector of non-negative Lagrange multipliers. Setting the derivative of (63) with respect to the primal variables ξ and $\boldsymbol{p}$ to zero yields, respectively,

$$\boldsymbol{\alpha}^\top\mathbf{1} = 1 \quad \text{and} \tag{64}$$

$$\boldsymbol{p} = -\boldsymbol{B}\boldsymbol{G}^{(i)}\boldsymbol{\alpha}. \tag{65}$$

The primal variable $\boldsymbol{p}$ and the dual variable $\boldsymbol{\alpha}$ are related via the dual connection (65). To eliminate the primal variables ξ and $\boldsymbol{p}$, we plug (64) and (65) back into the Lagrangian to obtain the dual of $M^{(i)}(\boldsymbol{p})$:

$$\begin{aligned} D^{(i)}(\boldsymbol{\alpha}) &:= -\tfrac{1}{2}(\boldsymbol{G}^{(i)}\boldsymbol{\alpha})^\top\boldsymbol{B}(\boldsymbol{G}^{(i)}\boldsymbol{\alpha}), \\ &\text{s.t. } \boldsymbol{\alpha}\in[0,1]^i,\ \|\boldsymbol{\alpha}\|_1 = 1. \end{aligned} \tag{66}$$

The dual objective $D^{(i)}(\boldsymbol{\alpha})$ (resp., primal objective $M^{(i)}(\boldsymbol{p})$) can be maximized (resp., minimized) exactly via quadratic programming. However, doing so may incur substantial computational expense. Instead we adopt an iterative scheme which is cheap and easy to implement yet guarantees dual improvement.

Let $\boldsymbol{\alpha}^{(i)} \in [0,1]^i$ be a feasible solution for $D^{(i)}(\boldsymbol{\alpha})$.[20] The corresponding primal solution $\boldsymbol{p}^{(i)}$ can be found by using (65). This in turn allows us to compute the next violating subgradient $\boldsymbol{g}^{(i+1)}$ via (61). With the new violating subgradient the dual becomes

$$\begin{aligned} D^{(i+1)}(\boldsymbol{\alpha}) &:= -\tfrac{1}{2}(\boldsymbol{G}^{(i+1)}\boldsymbol{\alpha})^\top\boldsymbol{B}(\boldsymbol{G}^{(i+1)}\boldsymbol{\alpha}), \\ &\text{s.t. } \boldsymbol{\alpha}\in[0,1]^{i+1},\ \|\boldsymbol{\alpha}\|_1 = 1, \end{aligned} \tag{67}$$

where the subgradient matrix is now extended:

$$\boldsymbol{G}^{(i+1)} = [\boldsymbol{G}^{(i)}, \boldsymbol{g}^{(i+1)}]. \tag{68}$$

Our iterative strategy constructs a new feasible solution $\boldsymbol{\alpha} \in [0,1]^{i+1}$ for (67) by constraining it to take the following form:

$$\boldsymbol{\alpha} = \begin{bmatrix} (1-\mu)\boldsymbol{\alpha}^{(i)} \\ \mu \end{bmatrix}, \quad \text{where } \mu\in[0,1]. \tag{69}$$

20. Note that $\boldsymbol{\alpha}^{(1)} = \mathbf{1}$ is a feasible solution for $D^{(1)}(\boldsymbol{\alpha})$.

In other words, we maximize a one-dimensional function $\bar{D}^{(i+1)} : [0,1] \to \mathbb{R}$:

$$\begin{aligned} \bar{D}^{(i+1)}(\mu) &:= -\tfrac{1}{2}\left(\boldsymbol{G}^{(i+1)}\boldsymbol{\alpha}\right)^\top \boldsymbol{B}\left(\boldsymbol{G}^{(i+1)}\boldsymbol{\alpha}\right) \\ &= -\tfrac{1}{2}\left((1-\mu)\bar{\boldsymbol{g}}^{(i)} + \mu\boldsymbol{g}^{(i+1)}\right)^\top \boldsymbol{B}\left((1-\mu)\bar{\boldsymbol{g}}^{(i)} + \mu\boldsymbol{g}^{(i+1)}\right), \end{aligned} \tag{70}$$

where

$$\bar{\boldsymbol{g}}^{(i)} := \boldsymbol{G}^{(i)}\boldsymbol{\alpha}^{(i)} \in \partial J(\boldsymbol{w}) \tag{71}$$

lies in the convex hull of $\boldsymbol{g}^{(j)} \in \partial J(\boldsymbol{w})\ \forall j \leq i$ (and hence in the convex set $\partial J(\boldsymbol{w})$) because $\boldsymbol{\alpha}^{(i)} \in [0,1]^i$ and $\|\boldsymbol{\alpha}^{(i)}\|_1 = 1$. Moreover, $\mu \in [0,1]$ ensures the feasibility of the dual solution. Noting that $\bar{D}^{(i+1)}(\mu)$ is a concave quadratic function, we set

$$\partial\bar{D}^{(i+1)}(\mu) = \left(\bar{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i+1)}\right)^\top \boldsymbol{B}\left((1-\eta)\bar{\boldsymbol{g}}^{(i)} + \eta\boldsymbol{g}^{(i+1)}\right) = 0 \tag{72}$$

to obtain the optimum

$$\mu^* := \operatorname*{argmax}_{\mu\in[0,1]} \bar{D}^{(i+1)}(\mu) = \min\left(1, \max\left(0, \frac{(\bar{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i+1)})^\top \boldsymbol{B}\bar{\boldsymbol{g}}^{(i)}}{(\bar{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i+1)})^\top \boldsymbol{B}(\bar{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i+1)})}\right)\right). \tag{73}$$

Our dual solution at step $i+1$ then becomes

$$\boldsymbol{\alpha}^{(i+1)} := \begin{bmatrix} (1-\mu^*)\boldsymbol{\alpha}^{(i)} \\ \mu^* \end{bmatrix}. \tag{74}$$

Furthermore, from (68), (69), and (71) it follows that $\bar{\boldsymbol{g}}^{(i)}$ can be maintained via an incremental update (Line 8 of Algorithm 2):

$$\bar{\boldsymbol{g}}^{(i+1)} := \boldsymbol{G}^{(i+1)}\boldsymbol{\alpha}^{(i+1)} = (1-\mu^*)\bar{\boldsymbol{g}}^{(i)} + \mu^*\boldsymbol{g}^{(i+1)}, \tag{75}$$

which combined with the dual connection (65) yields an incremental update for the primal solution (Line 9 of Algorithm 2):

$$\begin{aligned} \boldsymbol{p}^{(i+1)} := -\boldsymbol{B}\bar{\boldsymbol{g}}^{(i+1)} &= -(1-\mu^*)\boldsymbol{B}\bar{\boldsymbol{g}}^{(i)} - \mu^*\boldsymbol{B}\boldsymbol{g}^{(i+1)} \\ &= (1-\mu^*)\boldsymbol{p}^{(i)} - \mu^*\boldsymbol{B}\boldsymbol{g}^{(i+1)}. \end{aligned} \tag{76}$$

Using (75) and (76), computing a primal solution (Lines 7–9 of Algorithm 2) costs a total of $O(d^2)$ time (resp., $O(md)$ time for LBFGS with buffer size m), where d is the dimensionality of the optimization problem. Note that maximizing $D^{(i+1)}(\boldsymbol{\alpha})$ directly via quadratic programming generally results in a larger progress than that obtained by our approach.

In order to measure the quality of our solution at iteration i, we define the quantity

$$\epsilon^{(i)} := \min_{j\leq i} M^{(j+1)}(\boldsymbol{p}^{(j)}) - D^{(i)}(\boldsymbol{\alpha}^{(i)}) = \min_{j\leq i} M(\boldsymbol{p}^{(j)}) - D^{(i)}(\boldsymbol{\alpha}^{(i)}), \tag{77}$$

where the second equality follows directly from (62). Let $D(\boldsymbol{\alpha})$ be the corresponding dual problem of $M(\boldsymbol{p})$, with the property $D\left(\begin{bmatrix}\boldsymbol{\alpha}^{(i)} \\ \boldsymbol{0}\end{bmatrix}\right) = D^{(i)}(\boldsymbol{\alpha}^{(i)})$, and let $\boldsymbol{\alpha}^*$ be the optimal solution to

$\operatorname{argmax}_{\boldsymbol{\alpha}\in\mathcal{A}} D(\boldsymbol{\alpha})$ in some domain $\mathcal{A}$ of interest. As a consequence of the weak duality theorem (Hiriart-Urruty and Lemaréchal, 1993, Theorem XII.2.1.5), $\min_{\boldsymbol{p}\in\mathbb{R}^d} M(\boldsymbol{p}) \geq D(\boldsymbol{\alpha}^*)$. Therefore (77) implies that

$$\epsilon^{(i)} \;\geq\; \min_{\boldsymbol{p}\in\mathbb{R}^d} M(\boldsymbol{p}) - D^{(i)}(\boldsymbol{\alpha}^{(i)}) \;\geq\; \min_{\boldsymbol{p}\in\mathbb{R}^d} M(\boldsymbol{p}) - D(\boldsymbol{\alpha}^*) \;\geq\; 0. \tag{78}$$

The second inequality essentially says that $\epsilon^{(i)}$ is an upper bound on the duality gap. In fact, Theorem 7 below shows that $(\epsilon^{(i)} - \epsilon^{(i+1)})$ is bounded away from 0, that is, $\epsilon^{(i)}$ is monotonically decreasing. This guides us to design a practical stopping criterion (Line 6 of Algorithm 2) for our direction-finding procedure. Furthermore, using the dual connection (65), we can derive an implementable formula for $\epsilon^{(i)}$:

$$\begin{aligned}
\epsilon^{(i)} &= \min_{j\leq i}\left[\tfrac{1}{2}\,\boldsymbol{p}^{(j)\top}\boldsymbol{B}^{-1}\boldsymbol{p}^{(j)} + \boldsymbol{p}^{(j)\top}\boldsymbol{g}^{(j+1)} + \tfrac{1}{2}(\boldsymbol{G}^{(i)}\boldsymbol{\alpha}^{(i)})^\top\boldsymbol{B}(\boldsymbol{G}^{(i)}\boldsymbol{\alpha}^{(i)})\right] \\
&= \min_{j\leq i}\left[-\tfrac{1}{2}\,\boldsymbol{p}^{(j)\top}\bar{\boldsymbol{g}}^{(j)} + \boldsymbol{p}^{(j)\top}\boldsymbol{g}^{(j+1)} - \tfrac{1}{2}\,\boldsymbol{p}^{(i)\top}\bar{\boldsymbol{g}}^{(i)}\right] \\
&= \min_{j\leq i}\left[\boldsymbol{p}^{(j)\top}\boldsymbol{g}^{(j+1)} - \tfrac{1}{2}(\boldsymbol{p}^{(j)\top}\bar{\boldsymbol{g}}^{(j)} + \boldsymbol{p}^{(i)}{}^\top\bar{\boldsymbol{g}}^{(i)})\right], \\
&\quad \text{where } \boldsymbol{g}^{(j+1)} := \operatorname*{arg\,sup}_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p}^{(j)} \text{ and } \bar{\boldsymbol{g}}^{(j)} := \boldsymbol{G}^{(j)}\boldsymbol{\alpha}^{(j)}\ \ \forall j\leq i.
\end{aligned} \tag{79}$$

It is worth noting that continuous progress in the dual objective value does not necessarily prevent an increase in the primal objective value, that is, it is possible that $M(\boldsymbol{p}^{(i+1)}) \geq M(\boldsymbol{p}^{(i)})$. Therefore, we choose the best primal solution so far,

$$\boldsymbol{p} := \operatorname*{argmin}_{j\leq i} M(\boldsymbol{p}^{(j)}), \tag{80}$$

as the search direction (Line 18 of Algorithm 2) for the parameter update (3). This direction is a direction of descent as long as the last iterate $\boldsymbol{p}^{(i)}$ fulfills the descent condition (16). To see this, we use (88–90) below to get $\sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p}^{(i)} = M(\boldsymbol{p}^{(i)}) + D^{(i)}(\boldsymbol{\alpha}^{(i)})$, and since

$$M(\boldsymbol{p}^{(i)}) \geq \min_{j\leq i} M(\boldsymbol{p}^{(j)}) \text{ and } D^{(i)}(\boldsymbol{\alpha}^{(i)}) \geq D^{(j)}(\boldsymbol{\alpha}^{(j)})\ \ \forall j\leq i,$$

definition (80) immediately gives $\sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p}^{(i)} \geq \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p}$. Hence if $\boldsymbol{p}^{(i)}$ is a descent direction, then so is $\boldsymbol{p}$.

We now show that if the current parameter vector $\boldsymbol{w}$ is not optimal, then a direction-finding tolerance $\epsilon \geq 0$ exists for Algorithm 2 such that the returned search direction $\boldsymbol{p}$ is a descent direction, that is, $\sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p} < 0$.

Lemma 3 *Let $\boldsymbol{B}$ be the current approximation to the inverse Hessian maintained by Algorithm 1, and $h > 0$ a lower bound on the eigenvalues of $\boldsymbol{B}$. If the current iterate $\boldsymbol{w}$ is not optimal: $\mathbf{0} \notin \partial J(\boldsymbol{w})$, and the number of direction-finding iterations is unlimited ($k_{max} = \infty$), then there exists a direction-finding tolerance $\epsilon \geq 0$ such that the descent direction $\boldsymbol{p} = -\boldsymbol{B}\bar{\boldsymbol{g}}$, $\bar{\boldsymbol{g}} \in \partial J(\boldsymbol{w})$ returned by Algorithm 2 at $\boldsymbol{w}$ satisfies $\sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p} < 0$.*

Proof Algorithm 2 returns $\boldsymbol{p}$ after i iterations when $\epsilon^{(i)} \le \epsilon$, where $\epsilon^{(i)} = M(\boldsymbol{p}) - D^{(i)}(\boldsymbol{\alpha}^{(i)})$ by definitions (77) and (80). Using definition (66) of $D^{(i)}(\boldsymbol{\alpha}^{(i)})$, we have

$$-D^{(i)}(\boldsymbol{\alpha}^{(i)}) \;=\; \tfrac{1}{2}(\boldsymbol{G}^{(i)}\boldsymbol{\alpha}^{(i)})^\top \boldsymbol{B}(\boldsymbol{G}^{(i)}\boldsymbol{\alpha}^{(i)}) \;=\; \tfrac{1}{2}\,\bar{\boldsymbol{g}}^{(i)\top}\boldsymbol{B}\bar{\boldsymbol{g}}^{(i)}, \tag{81}$$

where $\bar{\boldsymbol{g}}^{(i)} = \boldsymbol{G}^{(i)}\boldsymbol{\alpha}^{(i)}$ is a subgradient in $\partial J(\boldsymbol{w})$. On the other hand, using (59) and (76), one can write

$$\begin{aligned} M(\boldsymbol{p}) \;&=\; \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p} \;+\; \tfrac{1}{2}\,\boldsymbol{p}^\top\boldsymbol{B}^{-1}\boldsymbol{p} \\ &=\; \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p} \;+\; \tfrac{1}{2}\,\bar{\boldsymbol{g}}^\top\boldsymbol{B}\bar{\boldsymbol{g}}, \quad \text{where} \quad \bar{\boldsymbol{g}} \in \partial J(\boldsymbol{w}). \end{aligned} \tag{82}$$

Putting together (81) and (82), and using $\boldsymbol{B} \succ h$, one obtains

$$\epsilon^{(i)} \;=\; \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p} \;+\; \tfrac{1}{2}\,\bar{\boldsymbol{g}}^\top\boldsymbol{B}\bar{\boldsymbol{g}} \;+\; \tfrac{1}{2}\,\bar{\boldsymbol{g}}^{(i)\top}\boldsymbol{B}\bar{\boldsymbol{g}}^{(i)} \;\ge\; \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p} \;+\; \frac{h}{2}\|\bar{\boldsymbol{g}}\|^2 \;+\; \frac{h}{2}\|\bar{\boldsymbol{g}}^{(i)}\|^2. \tag{83}$$

Since $\mathbf{0} \notin \partial J(\boldsymbol{w})$, the last two terms of (83) are strictly positive; and by (78), $\epsilon^{(i)} \ge 0$. The claim follows by choosing an ϵ such that $(\forall i)\;\; \frac{h}{2}(\|\bar{\boldsymbol{g}}\|^2 + \|\bar{\boldsymbol{g}}^{(i)}\|^2) > \epsilon \ge \epsilon^{(i)} \ge 0$. ■

Using the notation from Lemma 3, we show in the following corollary that a stricter upper bound on ϵ allows us to bound $\sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p}$ in terms of $\bar{\boldsymbol{g}}^\top\boldsymbol{B}\bar{\boldsymbol{g}}$ and $\|\bar{\boldsymbol{g}}\|$. This will be used in Appendix D to establish the global convergence of the subBFGS algorithm.

Corollary 4 *Under the conditions of Lemma 3, there exists an $\epsilon \ge 0$ for Algorithm 2 such that the search direction $\mathbf{p}$ generated by Algorithm 2 satisfies*

$$\sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p} \le -\tfrac{1}{2}\,\bar{\boldsymbol{g}}^\top\boldsymbol{B}\bar{\boldsymbol{g}} \;\le -\frac{h}{2}\|\bar{\boldsymbol{g}}\|^2 <\; 0. \tag{84}$$

Proof Using (83), we have

$$(\forall i)\;\; \epsilon^{(i)} \ge \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p} \;+\; \tfrac{1}{2}\,\bar{\boldsymbol{g}}^\top\boldsymbol{B}\bar{\boldsymbol{g}} \;+\; \frac{h}{2}\|\bar{\boldsymbol{g}}^{(i)}\|^2.$$

The first inequality in (84) results from choosing an ϵ such that

$$(\forall i)\;\; \frac{h}{2}\|\bar{\boldsymbol{g}}^{(i)}\|^2 \;\ge\; \epsilon \;\ge\; \epsilon^{(i)} \ge\; 0. \tag{85}$$

The lower bound $h > 0$ on the spectrum of $\boldsymbol{B}$ yields the second inequality in (84), and the third follows from the fact that $\|\bar{\boldsymbol{g}}\| > 0$ at non-optimal iterates. ■

Appendix B. Convergence of the Descent Direction Search

Using the notation established in Appendix A, we now prove the convergence of Algorithm 2 via several technical intermediate steps. The proof shares similarities with the proofs found in Smola et al. (2007), Shalev-Shwartz and Singer (2008), and Warmuth et al. (2008). The key idea is that at each iterate Algorithm 2 decreases the upper bound $\epsilon^{(i)}$ on the distance from the optimality, and the decrease in $\epsilon^{(i)}$ is characterized by the recurrence $\epsilon^{(i)} - \epsilon^{(i+1)} \geq c(\epsilon^{(i)})^2$ with $c > 0$ (Theorem 7). Analysing this recurrence then gives the convergence rate of the algorithm (Theorem 9).

We first provide two technical lemmas (Lemma 5 and 6) that are needed to prove Theorem 7.

Lemma 5 *Let $\bar{D}^{(i+1)}(\mu)$ be the one-dimensional function defined in (70), and $\epsilon^{(i)}$ the positive measure defined in (77). Then $\epsilon^{(i)} \leq \partial\bar{D}^{(i+1)}(0)$.*

Proof Let $\boldsymbol{p}^{(i)}$ be our primal solution at iteration i, derived from the dual solution $\boldsymbol{\alpha}^{(i)}$ using the dual connection (65). We then have

$$\boldsymbol{p}^{(i)} = -\boldsymbol{B}\bar{\boldsymbol{g}}^{(i)}, \quad \text{where} \quad \bar{\boldsymbol{g}}^{(i)} := \boldsymbol{G}^{(i)}\boldsymbol{\alpha}^{(i)}. \tag{86}$$

Definition (59) of $M(\boldsymbol{p})$ implies that

$$M(\boldsymbol{p}^{(i)}) \;=\; \tfrac{1}{2}\,\boldsymbol{p}^{(i)\top}\boldsymbol{B}^{-1}\boldsymbol{p}^{(i)} + \boldsymbol{p}^{(i)\top}\boldsymbol{g}^{(i+1)}, \tag{87}$$

where

$$\boldsymbol{g}^{(i+1)} \;:=\; \operatorname*{arg\,sup}_{\boldsymbol{g}\in\partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p}^{(i)}. \tag{88}$$

Using (86), we have $\boldsymbol{B}^{-1}\boldsymbol{p}^{(i)} = -\boldsymbol{B}^{-1}\boldsymbol{B}\bar{\boldsymbol{g}}^{(i)} = -\bar{\boldsymbol{g}}^{(i)}$, and hence (87) becomes

$$M(\boldsymbol{p}^{(i)}) \;=\; \boldsymbol{p}^{(i)\top}\boldsymbol{g}^{(i+1)} - \tfrac{1}{2}\,\boldsymbol{p}^{(i)\top}\bar{\boldsymbol{g}}^{(i)}. \tag{89}$$

Similarly, we have

$$D^{(i)}(\boldsymbol{\alpha}^{(i)}) \;=\; -\tfrac{1}{2}(\boldsymbol{G}^{(i)}\boldsymbol{\alpha}^{(i)})^\top\boldsymbol{B}(\boldsymbol{G}^{(i)}\boldsymbol{\alpha}^{(i)}) \;=\; \tfrac{1}{2}\,\boldsymbol{p}^{(i)\top}\bar{\boldsymbol{g}}^{(i)}. \tag{90}$$

From (72) and (86) it follows that

$$\partial\bar{D}^{(i+1)}(0) \;=\; (\bar{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i+1)})^\top\boldsymbol{B}\bar{\boldsymbol{g}}^{(i)} \;=\; (\boldsymbol{g}^{(i+1)} - \bar{\boldsymbol{g}}^{(i)})^\top\boldsymbol{p}^{(i)}, \tag{91}$$

where $\boldsymbol{g}^{(i+1)}$ is a violating subgradient chosen via (61), and hence coincides with (88). Using (89)–(91), we obtain

$$M(\boldsymbol{p}^{(i)}) - D^{(i)}(\boldsymbol{\alpha}^{(i)}) \;=\; \left(\boldsymbol{g}^{(i+1)} - \bar{\boldsymbol{g}}^{(i)}\right)^\top\boldsymbol{p}^{(i)} \;=\; \partial\bar{D}^{(i+1)}(0). \tag{92}$$

Together with definition (77) of $\epsilon^{(i)}$, (92) implies that

$$\begin{aligned}\epsilon^{(i)} \;&=\; \min_{j\leq i} M(\boldsymbol{p}^{(j)}) - D^{(i)}\left(\boldsymbol{\alpha}^{(i)}\right)\\ &\leq\; M(\boldsymbol{p}^{(i)}) - D^{(i)}(\boldsymbol{\alpha}^{(i)}) \;=\; \partial\bar{D}^{(i+1)}(0).\end{aligned}$$

■

Lemma 6 *Let $f : [0,1] \to \mathbb{R}$ be a concave quadratic function with $f(0) = 0$, $\partial f(0) \in [0,a]$, and $\partial f^2(x) \geq -a$ for some $a \geq 0$. Then $\max_{x\in[0,1]} f(x) \geq \frac{(\partial f(0))^2}{2a}$.*

Proof Using a second-order Taylor expansion around 0, we have $f(x) \geq \partial f(0)x - \frac{a}{2}x^2$. $x^* = \partial f(0)/a$ is the unconstrained maximum of the lower bound. Since $\partial f(0) \in [0,a]$, we have $x^* \in [0,1]$. Plugging x^* into the lower bound yields $(\partial f(0))^2/(2a)$. ■

Theorem 7 *Assume that at $\boldsymbol{w}$ the convex objective function $J : \mathbb{R}^d \to \mathbb{R}$ has bounded subgradient: $\|\partial J(\boldsymbol{w})\| \leq G$, and that the approximation $\boldsymbol{B}$ to the inverse Hessian has bounded eigenvalues: $\boldsymbol{B} \preceq H$. Then*

$$\epsilon^{(i)} - \epsilon^{(i+1)} \;\geq\; \frac{(\epsilon^{(i)})^2}{8G^2H}.$$

Proof Recall that we constrain the form of feasible dual solutions for $D^{(i+1)}(\boldsymbol{\alpha})$ as in (69). Instead of $D^{(i+1)}(\boldsymbol{\alpha})$, we thus work with the one-dimensional concave quadratic function $\bar{D}^{(i+1)}(\mu)$ (70). It is obvious that $\begin{bmatrix}\boldsymbol{\alpha}^{(i)}\\ 0\end{bmatrix}$ is a feasible solution for $D^{(i+1)}(\boldsymbol{\alpha})$. In this case, $\bar{D}^{(i+1)}(0) = D^{(i)}(\boldsymbol{\alpha}^{(i)})$. (74) implies that $\bar{D}^{(i+1)}(\mu^*) = D^{(i+1)}(\boldsymbol{\alpha}^{(i+1)})$. Using the definition (77) of $\epsilon^{(i)}$, we thus have

$$\epsilon^{(i)} - \epsilon^{(i+1)} \;\geq\; D^{(i+1)}(\boldsymbol{\alpha}^{(i+1)}) - D^{(i)}(\boldsymbol{\alpha}^{(i)}) \;=\; \bar{D}^{(i+1)}(\mu^*) - \bar{D}^{(i+1)}(0). \tag{93}$$

It is easy to see from (93) that $\epsilon^{(i)} - \epsilon^{(i+1)}$ are upper bounds on the maximal value of the concave quadratic function $f(\mu) := \bar{D}^{(i+1)}(\mu) - \bar{D}^{(i+1)}(0)$ with $\mu \in [0,1]$ and $f(0) = 0$. Furthermore, the definitions of $\bar{D}^{(i+1)}(\mu)$ and $f(\mu)$ imply that

$$\begin{aligned}
\partial f(0) \;&=\; \partial\bar{D}^{(i+1)}(0) \;=\; (\bar{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i+1)})^\top \boldsymbol{B}\bar{\boldsymbol{g}}^{(i)} \quad \text{and} \\
\partial^2 f(\mu) \;&=\; \partial^2\bar{D}^{(i+1)}(\mu) \;=\; -(\bar{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i+1)})^\top \boldsymbol{B}(\bar{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i+1)}).
\end{aligned} \tag{94}$$

Since $\|\partial J(\boldsymbol{w})\| \leq G$ and $\bar{\boldsymbol{g}}^{(i)} \in \partial J(\boldsymbol{w})$ (71), we have $\|\bar{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i+1)}\| \leq 2G$. Our upper bound on the spectrum of $\boldsymbol{B}$ then gives $|\partial f(0)| \leq 2G^2H$ and $\left|\partial^2 f(\mu)\right| \leq 4G^2H$. Additionally, Lemma 5 and the fact that $\boldsymbol{B} \succeq 0$ imply that

$$\partial f(0) \;=\; \partial\bar{D}^{(i+1)}(0) \geq 0 \quad \text{and} \quad \partial^2 f(\mu) \;=\; \partial^2\bar{D}^{(i+1)}(\mu) \leq 0, \tag{95}$$

which means that

$$\partial f(0) \;\in\; [0, 2G^2H] \subset [0, 4G^2H] \quad \text{and} \quad \partial^2 f(\mu) \;\geq\; -4G^2H.$$

Invoking Lemma 6, we immediately get

$$\epsilon^{(i)} - \epsilon^{(i+1)} \;\geq\; \frac{(\partial f(0))^2}{8G^2H} \;=\; \frac{(\partial\bar{D}^{(i+1)}(0))^2}{8G^2H}. \tag{96}$$

Since $\epsilon^{(i)} \leq \partial\bar{D}^{(i+1)}(0)$ by Lemma 5, the inequality (96) still holds when $\partial\bar{D}^{(i+1)}(0)$ is replaced with $\epsilon^{(i)}$. ■

(94) and (95) imply that the optimal combination coefficient μ^* (73) has the property

$$\mu^* = \min\left[1, \frac{\partial \bar{D}^{(i+1)}(0)}{-\partial^2 \bar{D}^{(i+1)}(\mu)}\right].$$

Moreover, we can use (65) to reduce the cost of computing μ^* by setting $\boldsymbol{B}\bar{\boldsymbol{g}}^{(i)}$ in (73) to be $-\boldsymbol{p}^{(i)}$ (Line 7 of Algorithm 2), and calculate

$$\mu^* = \min\left[1, \frac{\boldsymbol{g}^{(i+1)\top}\boldsymbol{p}^{(i)} - \bar{\boldsymbol{g}}^{(i)\top}\boldsymbol{p}^{(i)}}{\boldsymbol{g}^{(i+1)\top}\boldsymbol{B}_t\boldsymbol{g}^{(i+1)} + 2\,\boldsymbol{g}^{(i+1)\top}\boldsymbol{p}^{(i)} - \bar{\boldsymbol{g}}^{(i)\top}\boldsymbol{p}^{(i)}}\right], \tag{97}$$

where $\boldsymbol{B}_t\boldsymbol{g}^{(i+1)}$ can be cached for the update of the primal solution at Line 9 of Algorithm 2.

To prove Theorem 9, we use the following lemma proven by induction by Abe et al. (2001, Sublemma 5.4):

Lemma 8 *Let $\{\epsilon^{(1)}, \epsilon^{(2)}, \cdots\}$ be a sequence of non-negative numbers satisfying $\forall i \in \mathbb{N}$ the recurrence*

$$\epsilon^{(i)} - \epsilon^{(i+1)} \;\geq\; c\,(\epsilon^{(i)})^2,$$

where $c \in \mathbb{R}_+$ is a positive constant. Then $\forall i \in \mathbb{N}$ we have

$$\epsilon^{(i)} \;\leq\; \frac{1}{c\left(i + \frac{1}{\epsilon^{(1)}c}\right)}.$$

We now show that Algorithm 2 decreases $\epsilon^{(i)}$ to a pre-defined tolerance ϵ in $O(1/\epsilon)$ steps:

Theorem 9 *Under the assumptions of Theorem 7, Algorithm 2 converges to the desired precision ϵ after*

$$1 \;\leq\; t \;\leq\; \frac{8G^2H}{\epsilon} - 4$$

steps for any $\epsilon < 2G^2H$.

Proof Theorem 7 states that

$$\epsilon^{(i)} - \epsilon^{(i+1)} \;\geq\; \frac{(\epsilon^{(i)})^2}{8G^2H},$$

where $\epsilon^{(i)}$ is non-negative $\forall i \in \mathbb{N}$ by (78). Applying Lemma 8 we thus obtain

$$\epsilon^{(i)} \;\leq\; \frac{1}{c\left(i + \frac{1}{\epsilon^{(1)}c}\right)}, \quad \text{where} \quad c := \frac{1}{8G^2H}. \tag{98}$$

Our assumptions on $\|\partial J(\boldsymbol{w})\|$ and the spectrum of $\boldsymbol{B}$ imply that

$$\bar{D}^{(i+1)}(0) \;=\; (\bar{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i+1)})^\top \boldsymbol{B}\bar{\boldsymbol{g}}^{(i)} \;\leq\; 2G^2H.$$

Hence $\epsilon^{(i)} \leq 2G^2H$ by Lemma 5. This means that (98) holds with $\epsilon^{(1)} = 2G^2H$. Therefore we can solve

$$\epsilon \leq \frac{1}{c\left(t + \frac{1}{\epsilon^{(1)}c}\right)} \quad \text{with} \quad c := \frac{1}{8G^2H} \quad \text{and} \quad \epsilon^{(1)} := 2G^2H \tag{99}$$

to obtain an upper bound on t such that $(\forall i \geq t)\ \epsilon^{(i)} \leq \epsilon < 2G^2H$. The solution to (99) is $t \leq \frac{8G^2H}{\epsilon} - 4$. ■

Appendix C. Satisfiability of the Subgradient Wolfe Conditions

To formally show that there always is a positive step size that satisfies the subgradient Wolfe conditions (23, 24), we restate a result of Hiriart-Urruty and Lemaréchal (1993, Theorem VI.2.3.3) in slightly modified form:

Lemma 10 *Given two points $\boldsymbol{w} \neq \boldsymbol{w}'$ in $\mathbb{R}^d$, define $\boldsymbol{w}_\eta = \eta\boldsymbol{w}' + (1-\eta)\boldsymbol{w}$. Let $J : \mathbb{R}^d \to \mathbb{R}$ be convex. There exists $\eta \in (0,1)$ and $\tilde{\boldsymbol{g}} \in \partial J(\boldsymbol{w}_\eta)$ such that*

$$J(\boldsymbol{w}') - J(\boldsymbol{w}) \;=\; \tilde{\boldsymbol{g}}^\top(\boldsymbol{w}' - \boldsymbol{w}) \;\leq\; \hat{\boldsymbol{g}}^\top(\boldsymbol{w}' - \boldsymbol{w}),$$

where $\hat{\boldsymbol{g}} := \arg\sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_\eta)} \boldsymbol{g}^\top(\boldsymbol{w}' - \boldsymbol{w})$.

Theorem 11 *Let $\boldsymbol{p}$ be a descent direction at an iterate $\boldsymbol{w}$. If $\Phi(\eta) := J(\boldsymbol{w} + \eta\boldsymbol{p})$ is bounded below, then there exists a step size $\eta > 0$ which satisfies the subgradient Wolfe conditions (23, 24).*

Proof Since $\boldsymbol{p}$ is a descent direction, the line $J(\boldsymbol{w}) + c_1\eta \sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p}$ with $c_1 \in (0,1)$ must intersect $\Phi(\eta)$ at least once at some $\eta > 0$ (see Figure 1 for geometric intuition). Let η' be the smallest such intersection point; then

$$J(\boldsymbol{w} + \eta'\boldsymbol{p}) \;=\; J(\boldsymbol{w}) \;+\; c_1\eta' \sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p}. \tag{100}$$

Since $\Phi(\eta)$ is lower bounded, the sufficient decrease condition (23) holds for all $\eta'' \in [0, \eta']$. Setting $\boldsymbol{w}' = \boldsymbol{w} + \eta'\boldsymbol{p}$ in Lemma 10 implies that there exists an $\eta'' \in (0, \eta')$ such that

$$J(\boldsymbol{w} + \eta'\boldsymbol{p}) \;-\; J(\boldsymbol{w}) \;\leq\; \eta' \sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w} + \eta''\boldsymbol{p})} \boldsymbol{g}^\top\boldsymbol{p}. \tag{101}$$

Plugging (100) into (101) and simplifying it yields

$$c_1 \sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p} \;\leq\; \sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w} + \eta''\boldsymbol{p})} \boldsymbol{g}^\top\boldsymbol{p}. \tag{102}$$

Since $\boldsymbol{p}$ is a descent direction, $\sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w})} \boldsymbol{g}^\top\boldsymbol{p} < 0$, and thus (102) also holds when c_1 is replaced by $c_2 \in (c_1, 1)$. ■

Algorithm 6 Algorithm 1 of Birge et al. (1998)

Initialize: $t := 0$ and $\boldsymbol{w}_0$

while not converged **do**

Find $\boldsymbol{w}_{t+1}$ that obeys

$$J(\boldsymbol{w}_{t+1}) \;\leq\; J(\boldsymbol{w}_t) \;-\; a_t \,\|\boldsymbol{g}_{\epsilon_t'}\|^2 + \epsilon_t \qquad (104)$$
$$\text{where } \; \boldsymbol{g}_{\epsilon_t'} \in \partial_{\epsilon_t'} J(\boldsymbol{w}_{t+1}),\; a_t > 0,\; \epsilon_t, \epsilon_t' \geq 0\,.$$

$t := t + 1$

end while

Appendix D. Global Convergence of SubBFGS

There are technical difficulties in extending the classical BFGS convergence proof to the nonsmooth case. This route was taken by Andrew and Gao (2007), which unfortunately left their proof critically flawed: In a key step (Andrew and Gao, 2007, Equation 7) they seek to establish the non-negativity of the directional derivative $f'(\bar{x};\bar{q})$ of a convex function f at a point $\bar{x}$ in the direction $\bar{q}$, where $\bar{x}$ and $\bar{q}$ are the limit points of convergent sequences $\{x^k\}$ and $\{\hat{q}^k\}_\kappa$, respectively. They do so by taking the limit for $k \in \kappa$ of

$$f'(x^k + \tilde{\alpha}^k \hat{q}^k; \hat{q}^k) \;>\; \gamma f'(x^k; \hat{q}^k), \;\; \text{where} \;\; \{\tilde{\alpha}^k\} \to 0 \;\; \text{and} \;\; \gamma \in (0, 1)\,,$$

which leads them to claim that

$$f'(\bar{x};\bar{q}) \;\geq\; \gamma f'(\bar{x};\bar{q})\,, \qquad (103)$$

which would imply $f'(\bar{x};\bar{q}) \geq 0$ because $\gamma \in (0,1)$. However, $f'(x^k, \hat{q}^k)$ does not necessarily converge to $f'(\bar{x};\bar{q})$ because the directional derivative of a nonsmooth convex function is not continuous, only *upper semi-continuous* (Bertsekas, 1999, Proposition B.23). Instead of (103) we thus only have

$$f'(\bar{x};\bar{q}) \;\geq\; \gamma \limsup_{k\to\infty, k\in\kappa} f'(x^k; \hat{q}^k)\,,$$

which does not suffice to establish the desired result: $f'(\bar{x};\bar{q}) \geq 0$. A similar mistake is also found in the reasoning of Andrew and Gao (2007) just after Equation 7.

Instead of this flawed approach, we use the technique introduced by Birge et al. (1998) to prove the global convergence of subBFGS (Algorithm 1) in objective function value, that is, $J(\boldsymbol{w}_t) \to \inf_{\boldsymbol{w}} J(\boldsymbol{w})$, provided that the spectrum of BFGS' inverse Hessian approximation $\boldsymbol{B}_t$ is bounded above and below for all t, and the step size η_t (obtained at Line 9) is not summable: $\sum_{t=0}^{\infty} \eta_t = \infty$.

Birge et al. (1998) provide a unified framework for convergence analysis of optimization algorithms for nonsmooth convex optimization, based on the notion of *ϵ-subgradients*. Formally, $\boldsymbol{g}$ is called an ϵ-subgradient of J at $\boldsymbol{w}$ iff (Hiriart-Urruty and Lemaréchal, 1993, Definition XI.1.1.1)

$$(\forall \boldsymbol{w}') \;\; J(\boldsymbol{w}') \;\geq\; J(\boldsymbol{w}) + (\boldsymbol{w}' - \boldsymbol{w})^\top \boldsymbol{g} - \epsilon, \;\; \text{where} \;\; \epsilon \geq 0. \qquad (105)$$

The set of all ϵ-subgradients at a point $\boldsymbol{w}$ is called the ϵ-subdifferential, and denoted $\partial_\epsilon J(\boldsymbol{w})$. From the definition of subgradient (7), it is easy to see that $\partial J(\boldsymbol{w}) = \partial_0 J(\boldsymbol{w}) \subseteq \partial_\epsilon J(\boldsymbol{w})$. Birge et al. (1998) propose an ϵ-subgradient-based algorithm (Algorithm 6) and provide sufficient conditions for its global convergence:

Theorem 12 (Birge et al., 1998, Theorem 2.1(iv), first sentence)
Let $J : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$ be a proper lower semi-continuous[21] *extended-valued convex function, and let $\{(\epsilon_t, \epsilon'_t, a_t, \boldsymbol{w}_{t+1}, \boldsymbol{g}_{\epsilon'_t})\}$ be any sequence generated by Algorithm 6 satisfying*

$$\sum_{t=0}^{\infty} \epsilon_t < \infty \quad and \quad \sum_{t=0}^{\infty} a_t = \infty. \tag{106}$$

If $\epsilon'_t \to 0$, and there exists a positive number $\beta > 0$ such that, for all large t,

$$\beta \, \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\| \;\leq\; a_t \|\boldsymbol{g}_{\epsilon'_t}\|, \tag{107}$$

then $J(\boldsymbol{w}_t) \to \inf_{\boldsymbol{w}} J(\boldsymbol{w})$.

We will use this result to establish the global convergence of subBFGS in Theorem 14. Towards this end, we first show that subBFGS is a special case of Algorithm 6:

Lemma 13 *Let $\boldsymbol{p}_t = -\boldsymbol{B}_t \bar{\boldsymbol{g}}_t$ be the descent direction produced by Algorithm 2 at a non-optimal iterate $\boldsymbol{w}_t$, where $\boldsymbol{B}_t \succeq h > 0$ and $\bar{\boldsymbol{g}}_t \in \partial J(\boldsymbol{w}_t)$, and let $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta_t \boldsymbol{p}_t$, where $\eta_t > 0$ satisfies sufficient decrease (23) with free parameter $c_1 \in (0,1)$. Then $\boldsymbol{w}_{t+1}$ obeys (104) of Algorithm 6 for $a_t := \frac{c_1 \eta_t h}{2}$, $\epsilon_t = 0$, and $\epsilon'_t := \eta_t(1 - \frac{c_1}{2})\, \bar{\boldsymbol{g}}_t^\top \boldsymbol{B}_t \bar{\boldsymbol{g}}_t$.*

Proof Our sufficient decrease condition (23) and Corollary 4 imply that

$$\begin{aligned} J(\boldsymbol{w}_{t+1}) \;&\leq\; J(\boldsymbol{w}_t) \;-\; \frac{c_1 \eta_t}{2}\, \bar{\boldsymbol{g}}_t^\top \boldsymbol{B}_t \bar{\boldsymbol{g}}_t \\ &\leq\; J(\boldsymbol{w}_t) \;-\; a_t \|\bar{\boldsymbol{g}}_t\|^2, \;\; \text{where} \;\; a_t := \frac{c_1 \eta_t h}{2}. \end{aligned} \tag{108}$$

What is left to prove is that $\bar{\boldsymbol{g}}_t \in \partial_{\epsilon'_t} J(\boldsymbol{w}_{t+1})$ for an $\epsilon'_t \geq 0$. Using $\bar{\boldsymbol{g}}_t \in \partial J(\boldsymbol{w}_t)$ and the definition (7) of subgradient, we have

$$\begin{aligned} (\forall \boldsymbol{w}) \;\; J(\boldsymbol{w}) \;&\geq\; J(\boldsymbol{w}_t) \;+\; (\boldsymbol{w} - \boldsymbol{w}_t)^\top \bar{\boldsymbol{g}}_t \\ &=\; J(\boldsymbol{w}_{t+1}) \;+\; (\boldsymbol{w} - \boldsymbol{w}_{t+1})^\top \bar{\boldsymbol{g}}_t \;+\; J(\boldsymbol{w}_t) - J(\boldsymbol{w}_{t+1}) \;+\; (\boldsymbol{w}_{t+1} - \boldsymbol{w}_t)^\top \bar{\boldsymbol{g}}_t \,. \end{aligned}$$

Using $\boldsymbol{w}_{t+1} - \boldsymbol{w}_t = -\eta_t \boldsymbol{B}_t \bar{\boldsymbol{g}}_t$ and (108) gives

$$\begin{aligned} (\forall \boldsymbol{w}) \;\; J(\boldsymbol{w}) \;&\geq\; J(\boldsymbol{w}_{t+1}) \;+\; (\boldsymbol{w} - \boldsymbol{w}_{t+1})^\top \bar{\boldsymbol{g}}_t \;+\; \frac{c_1 \eta_t}{2}\, \bar{\boldsymbol{g}}_t^\top \boldsymbol{B}_t \bar{\boldsymbol{g}}_t \;-\; \eta_t\, \bar{\boldsymbol{g}}_t^\top \boldsymbol{B}_t \bar{\boldsymbol{g}}_t \\ &=\; J(\boldsymbol{w}_{t+1}) \;+\; (\boldsymbol{w} - \boldsymbol{w}_{t+1})^\top \bar{\boldsymbol{g}}_t \;-\; \epsilon'_t \,, \end{aligned}$$

where $\epsilon'_t := \eta_t(1 - \frac{c_1}{2})\, \bar{\boldsymbol{g}}_t^\top \boldsymbol{B}_t \bar{\boldsymbol{g}}_t$. Since $\eta_t > 0$, $c_1 < 1$, and $\boldsymbol{B}_t \succeq h > 0$, ϵ'_t is non-negative. By the definition (105) of ϵ-subgradient, $\bar{\boldsymbol{g}}_t \in \partial_{\epsilon'_t} J(\boldsymbol{w}_{t+1})$. ■

21. This means that there exists at least one $\boldsymbol{w} \in \mathbb{R}^d$ such that $J(\boldsymbol{w}) < \infty$, and that for all $\boldsymbol{w} \in \mathbb{R}^d$, $J(\boldsymbol{w}) > -\infty$ and $J(\boldsymbol{w}) \leq \liminf_{t \to \infty} J(\boldsymbol{w}_t)$ for any sequence $\{\boldsymbol{w}_t\}$ converging to $\boldsymbol{w}$. All objective functions considered in this paper fulfill these conditions.

Theorem 14 *Let $J : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$ be a proper lower semi-continuous[21] extended-valued convex function. Algorithm 1 with a line search that satisfies the sufficient decrease condition (23) with $c_1 \in (0,1)$ converges globally to the minimal value of J, provided that:*

1. *the spectrum of its approximation to the inverse Hessian is bounded above and below: $\exists\, (h, H : 0 < h \leq H < \infty) : (\forall t)\ h \preceq \boldsymbol{B}_t \preceq H$*
2. *the step size $\eta_t > 0$ satisfies $\sum_{t=0}^{\infty} \eta_t = \infty$, and*
3. *the direction-finding tolerance ϵ for Algorithm 2 satisfies (85).*

Proof We have already shown in Lemma 13 that subBFGS is a special case of Algorithm 6. Thus if we can show that the technical conditions of Theorem 12 are met, it directly establishes the global convergence of subBFGS.

Recall that for subBFGS $a_t := \frac{c_1 \eta_t h}{2}$, $\epsilon_t = 0$, $\epsilon'_t := \eta_t(1 - \frac{c_1}{2})\, \bar{\boldsymbol{g}}_t^\top \boldsymbol{B}_t \bar{\boldsymbol{g}}_t$, and $\bar{\boldsymbol{g}}_t = \boldsymbol{g}_{\epsilon'_t}$. Our assumption on η_t implies that $\sum_{t=0}^{\infty} a_t = \frac{c_1 h}{2} \sum_{t=0}^{\infty} \eta_t = \infty$, thus establishing (106). We now show that $\epsilon'_t \to 0$. Under the third condition of Theorem 14, it follows from the first inequality in (84) in Corollary 4 that

$$\sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}_t \ \leq\ -\tfrac{1}{2}\, \bar{\boldsymbol{g}}_t^\top \boldsymbol{B}_t \bar{\boldsymbol{g}}_t\,, \tag{109}$$

where $\boldsymbol{p}_t = -\boldsymbol{B}_t \bar{\boldsymbol{g}}_t,\ \bar{\boldsymbol{g}}_t \in \partial J(\boldsymbol{w}_t)$ is the search direction returned by Algorithm 2. Together with the sufficient decrease condition (23), (109) implies (108). Now use (108) recursively to obtain

$$J(\boldsymbol{w}_{t+1}) \ \leq\ J(\boldsymbol{w}_0) \ -\ \frac{c_1}{2} \sum_{i=0}^{t} \eta_i\, \bar{\boldsymbol{g}}_i^\top \boldsymbol{B}_i \bar{\boldsymbol{g}}_i\,.$$

Since J is proper (hence bounded from below), we have

$$\sum_{t=0}^{\infty} \eta_i\, \bar{\boldsymbol{g}}_i^\top \boldsymbol{B}_i \bar{\boldsymbol{g}}_i \ =\ \frac{1}{1 - \frac{c_1}{2}} \sum_{t=0}^{\infty} \epsilon'_i \ <\ \infty\,. \tag{110}$$

Recall that $\epsilon'_i \geq 0$. The bounded sum of non-negative terms in (110) implies that the terms in the sum must converge to zero.

Finally, to show (107) we use $\boldsymbol{w}_{t+1} - \boldsymbol{w}_t = -\eta_t \boldsymbol{B}_t \bar{\boldsymbol{g}}_t$, the definition of the matrix norm: $\|\boldsymbol{B}\| := \max_{\boldsymbol{x} \neq 0} \frac{\|\boldsymbol{B}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}$, and the upper bound on the spectrum of $\boldsymbol{B}_t$ to write:

$$\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\| \ =\ \eta_t \|\boldsymbol{B}_t \bar{\boldsymbol{g}}_t\| \ \leq\ \eta_t \|\boldsymbol{B}_t\| \|\bar{\boldsymbol{g}}_t\| \ \leq\ \eta_t H \|\bar{\boldsymbol{g}}_t\|. \tag{111}$$

Recall that $\bar{\boldsymbol{g}}_t = \boldsymbol{g}_{\epsilon'_t}$ and $a_t = \frac{c_1 \eta_t h}{2}$, and multiply both sides of (111) by $\frac{c_1 h}{2H}$ to obtain (107) with $\beta := \frac{c_1 h}{2H}$. ■

Appendix E. SubBFGS Converges on Various Counterexamples

We demonstrate the global convergence of subBFGS[22] with an exact line search on various counterexamples from the literature, designed to show the failure to converge of other gradient-based algorithms.

22. We run Algorithm 1 with $h = 10^{-8}$ and $\epsilon = 10^{-5}$.

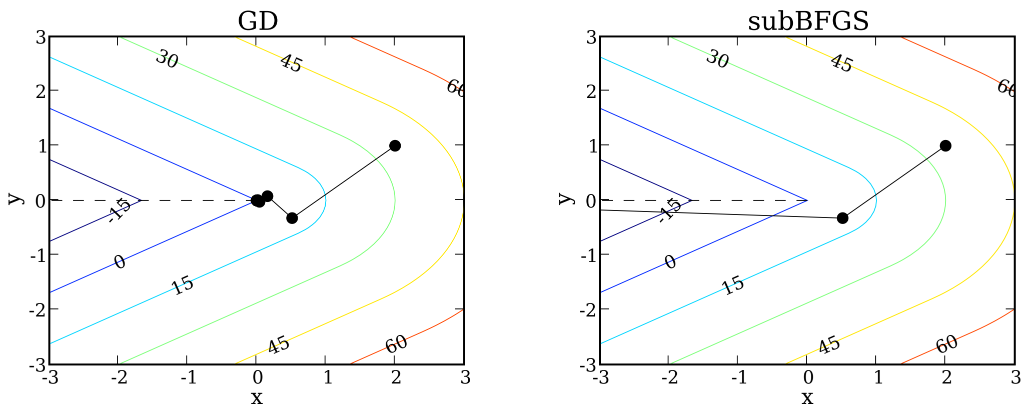


Figure 23: Optimization trajectory of steepest descent (left) and subBFGS (right) on counterexample (112).

E.1 Counterexample for Steepest Descent

The first counterexample (112) is given by Wolfe (1975) to show the non-convergent behaviour of the steepest descent method with an exact line search (denoted GD):

$$f(x,y) := \begin{cases} 5\sqrt{(9x^2+16y^2)} & \text{if} \quad x \geq |y|, \\ 9x+16|y| & \text{otherwise}. \end{cases} \tag{112}$$

This function is subdifferentiable along $x \leq 0,\ y = 0$ (dashed line in Figure 23); its minimal value ($-\infty$) is attained for $x = -\infty$. As can be seen in Figure 23 (left), starting from a differentiable point $(2,1)$, GD follows successively orthogonal directions, that is, $-\nabla f(x,y)$, and converges to the non-optimal point $(0,0)$. As pointed out by Wolfe (1975), the failure of GD here is due to the fact that GD does not have a global view of f, specifically, it is because the gradient evaluated at each iterate (solid disk) is not informative about $\partial f(0,0)$, which contains subgradients (e.g., $(9,0)$), whose negative directions point toward the minimum. SubBFGS overcomes this "short-sightedness" by incorporating into the parameter update (3) an estimate $\boldsymbol{B}_t$ of the inverse Hessian, whose information about the shape of f prevents subBFGS from zigzagging to a non-optimal point. Figure 23 (right) shows that subBFGS moves to the correct region ($x < 0$) at the second step. In fact, the second step of subBFGS lands exactly on the hinge $x \leq 0, y = 0$, where a subgradient pointing to the optimum is available.

E.2 Counterexample for Steepest Subgradient Descent

The second counterexample (113), due to Hiriart-Urruty and Lemaréchal (1993, Section VIII.2.2), is a piecewise linear function which is subdifferentiable along $0 \leq y = \pm 3x$ and $x = 0$ (dashed lines in Figure 24):

$$f(x,y) := \max\{-100,\ \pm 2x + 3y,\ \pm 5x + 2y\}. \tag{113}$$

This example shows that steepest subgradient descent with an exact line search (denoted subGD) may not converge to the optimum of a nonsmooth function. Steepest subgradient descent updates

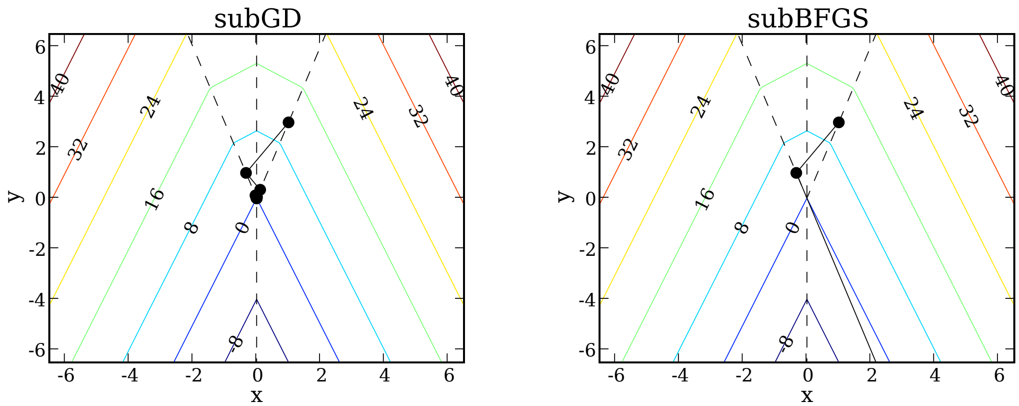


Figure 24: Optimization trajectory of steepest subgradient descent (left) and subBFGS (right) on counterexample (113).

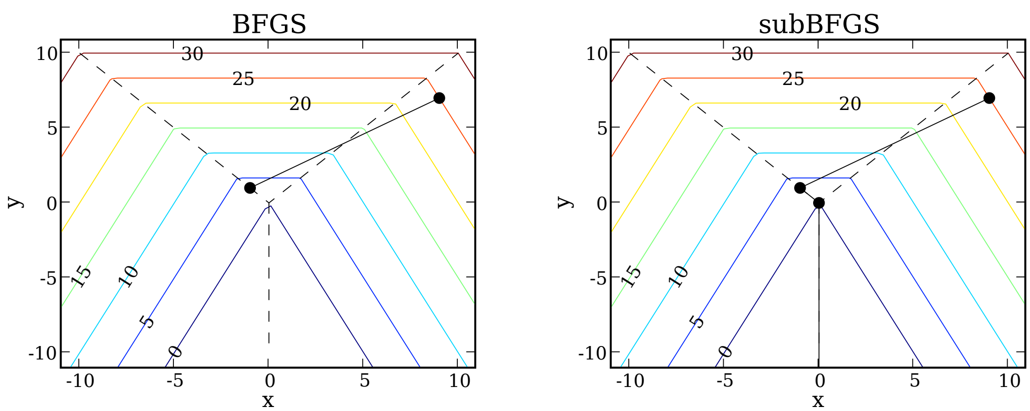


Figure 25: Optimization trajectory of standard BFGS (left) and subBFGS (right) on counterexample (114).

parameters along the *steepest descent* subgradient direction, which is obtained by solving the min-sup problem (13) with respect to the Euclidean norm. Clearly, the minimal value of f (-100) is attained for sufficiently negative values of y. However, subGD oscillates between two hinges $0 \le y = \pm 3x$, converging to the non-optimal point $(0,0)$, as shown in Figure 24 (left). The zigzagging optimization trajectory of subGD does not allow it to land on any informative position such as the hinge $y = 0$, where the steepest subgradient descent direction points to the desired region $(y < 0)$; Hiriart-Urruty and Lemaréchal (1993, Section VIII.2.2) provide a detailed discussion. By contrast, subBFGS moves to the $y < 0$ region at the second step (Figure 24, right), which ends at the point $(100, -300)$ (not shown in the figure) where the minimal value of f is attained.

E.3 Counterexample for BFGS

The final counterexample (114) is given by Lewis and Overton (2008b) to show that the standard BFGS algorithm with an exact line search can break down when encountering a nonsmooth point:

$$f(x, y) := \max\{2|x| + y,\ 3y\}. \tag{114}$$

This function is subdifferentiable along $x = 0,\ y \leq 0$ and $y = |x|$ (dashed lines in Figure 25). Figure 25 (left) shows that after the first step, BFGS lands on a nonsmooth point, where it fails to find a descent direction. This is not surprising because at a nonsmooth point $\boldsymbol{w}$ the quasi-Newton direction $\boldsymbol{p} := -\boldsymbol{B}\boldsymbol{g}$ for a given subgradient $\boldsymbol{g} \in \partial J(\boldsymbol{w})$ is not necessarily a direction of descent. SubBFGS fixes this problem by using a direction-finding procedure (Algorithm 2), which is guaranteed to generate a descent quasi-Newton direction. Here subBFGS converges to $f = -\infty$ in three iterations (Figure 25, right).

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.

P. K. Agarwal and M. Sharir. Davenport-Schinzel sequences and their geometric applications. In J. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 1–47. North-Holland, New York, 2000.

G. Andrew and J. Gao. Scalable training of L_1-regularized log-linear models. In *Proc. Intl. Conf. Machine Learning*, pages 33–40, New York, NY, USA, 2007. ACM.

J. Basch. *Kinetic Data Structures*. PhD thesis, Stanford University, June 1999.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.

J. R. Birge, L. Qi, and Z. Wei. A general approach to convergence properties of some methods for nonsmooth convex optimization. *Applied Mathematics and Optimization*, 38(2):141–158, 1998.

A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with LaRank. In *Proc. Intl. Conf. Machine Learning*, pages 89–96, New York, NY, USA, 2007. ACM.

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

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

K. Crammer and Y. Singer. A family of additive online algorithms for category ranking. *J. Mach. Learn. Res.*, 3:1025–1058, February 2003b.

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. In A. McCallum and S. Roweis, editors, *ICML*, pages 320–327. Omnipress, 2008.

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for large-scale risk minimization. *Journal of Machine Learning Research*, 10:2157–2192, 2009.

M. Haarala. *Large-Scale Nonsmooth Optimization*. PhD thesis, University of Jyväskylä, 2004.

J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Information Processing Letters*, 33(4):169–174, December 1989.

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

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

Y. J. Lee and O. L. Mangasarian. SSVM: A smooth support vector machine for classification. *Computational optimization and Applications*, 20(1):5–22, 2001.

C. Lemarechal. Numerical experiments in nonsmooth optimization. *Progress in Nondifferentiable Optimization*, 82:61–84, 1982.

A. S. Lewis and M. L. Overton. Nonsmooth optimization via BFGS. Technical report, Optimization Online, 2008a. http://www.optimization-online.org/DB_FILE/2008/12/2172.pdf, submitted to SIAM J. Optimization.

A. S. Lewis and M. L. Overton. Behavior of BFGS with an exact line search on nonsmooth examples. Technical report, Optimization Online, 2008b. http://www.optimization-online.org/DB_FILE/2008/12/2173.pdf, submitted to SIAM J. Optimization.

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989.

L. Lukšan and J. Vlček. Globally convergent variable metric method for convex nonsmooth unconstrained minimization. *Journal of Optimization Theory and Applications*, 102(3):593–613, 1999.

F. Maes, L. Denoyer, and P. Gallinari. XML structure mapping application to the PASCAL/INEX 2006 XML document mining track. In *Advances in XML Information Retrieval and Evaluation: Fifth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX'06)*, Dagstuhl, Germany, 2007.

A. Nedić 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.

A. Nemirovski. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM J. on Optimization*, 15(1):229–251, 2005. ISSN 1052-6234.

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

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

S. Shalev-Shwartz and Y. Singer. On the equivalence of weak learnability and linear separability: New relaxations and efficient boosting algorithms. In *Proceedings of COLT*, 2008.

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.

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, S. V. N. Vishwanthan, A. J. Smola, and Q. V. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11:311–365, 2010.

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.

M. K. Warmuth, K. A. Glocer, and S. V. N. Vishwanathan. Entropy regularized LPBoost. In Y. Freund, Y. Làszlò Györfi, and G. Turàn, editors, *Proc. Intl. Conf. Algorithmic Learning Theory*, number 5254 in Lecture Notes in Artificial Intelligence, pages 256 – 271, Budapest, October 2008. Springer-Verlag.

P. Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969.

P. Wolfe. A method of conjugate subgradients for minimizing nondifferentiable functions. *Mathematical Programming Study*, 3:145–173, 1975.

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 and F. J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4:5–31, 2001.