A Quasi-Newton Approach to Nonsmooth Convex Optimization

Jin Yu JIN.YU@ANU.EDU.AU
S.V. N. Vishwanathan SVN.VISHWANATHAN@NICTA.COM.AU
Simon Günter GUENTER_SIMON@HOTMAIL.COM
Nicol N. Schraudolph NIC@SCHRAUDOLPH.ORG
Canberra Research Laboratory, NICTA, Canberra, Australia
Research School of Information Sciences & Engineering, Australian National University, Canberra, Australia

Abstract

We extend the well-known BFGS quasi-Newton method and its limited-memory 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 apply the resulting subLBFGS algorithm to L_2-regularized risk minimization with binary hinge loss, and its direction-finding component to L_1-regularized risk minimization with logistic loss. In both settings our generic algorithms perform comparable to or better than their counterparts in specialized state-of-the-art solvers.

1. Introduction

The (L)BFGS quasi-Newton method (Nocedal and Wright, 1999) is widely regarded as the workhorse of smooth nonlinear optimization due to its combination of computational efficiency with good asymptotic convergence. Given a smooth objective 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:

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

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

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

which is used for the parameter update:

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

The step size $\eta_t \in \mathbb{R}_+$ is normally determined by a line search obeying the Wolfe conditions:

$$\begin{aligned} J(\boldsymbol{w}_{t+1}) \ &\leq \ J(\boldsymbol{w}_t) + c_1 \eta_t \nabla J(\boldsymbol{w}_t)^\top \boldsymbol{p}_t \\ \text{and} \quad \nabla J(\boldsymbol{w}_{t+1})^\top \boldsymbol{p}_t \ &\geq \ c_2 \nabla J(\boldsymbol{w}_t)^\top \boldsymbol{p}_t, \end{aligned} \qquad (4)$$

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

$$\boldsymbol{B}_{t+1} = (\boldsymbol{I} - \varrho_t \boldsymbol{s}_t \boldsymbol{y}_t^\top)\boldsymbol{B}_t(\boldsymbol{I} - \varrho_t \boldsymbol{y}_t \boldsymbol{s}_t^\top) + \varrho_t \boldsymbol{s}_t \boldsymbol{s}_t^\top, \qquad (5)$$

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 $\varrho_t := (\boldsymbol{y}_t^\top \boldsymbol{s}_t)^{-1}$. 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) is a variant of BFGS designed for solving large-scale optimization problems where the $O(d^2)$ cost of storing and updating $\boldsymbol{B}_t$ would be prohibitively expensive. LBFGS approximates the quasi-Newton direction directly from the last m pairs of $\boldsymbol{s}_t$ and $\boldsymbol{y}_t$ via a matrix-free approach. This reduces the cost to $O(md)$ space and time per iteration, with m freely chosen (Nocedal and Wright, 1999).

Smoothness of the objective function is essential for standard (L)BFGS because both the local quadratic model (1) and the Wolfe conditions (4) require the existence of the gradient ∇J at every point. Even though nonsmooth convex functions are differentiable everywhere except on a set of Lebesgue measure zero (Hiriart-Urruty and Lemaréchal, 1993), in practice (L)BFGS often fails to converge on such problems (Lukšan and Vlček, 1999; Haarala, 2004). Various subgradient-based approaches, such as subgradient descent (Nedich and Bertsekas, 2000) or bundle methods (Teo et al., 2007), are therefore preferred.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Although a convex function might not be differentiable everywhere, a subgradient always exists. 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

$$J(\boldsymbol{w}') \;\geq\; J(\boldsymbol{w}) + (\boldsymbol{w}' - \boldsymbol{w})^\top \boldsymbol{g} \quad \forall \boldsymbol{w}'. \tag{6}$$

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

In this paper we systematically modify the standard (L)BFGS algorithm so as to make it amenable to subgradients. This results in sub(L)BFGS, a new subgradient quasi-Newton method which is applicable to a wide variety of nonsmooth convex optimization problems encountered in machine learning.

In the next section we describe our new algorithm generically, before we discuss its application to L_2-regularized risk minimization with hinge loss in Section 3. Section 4 compares and contrasts our work with other recent efforts in this area. Encouraging experimental results are reported in Section 5. We conclude with an outlook and discussion in Section 6.

2. Subgradient BFGS Method

We modify the standard BFGS algorithm to derive our new algorithm (subBFGS, Algorithm 1) for nonsmooth convex optimization. These 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.

2.1. Generalizing the Local Quadratic Model

Recall that BFGS assumes the objective function J is differentiable everywhere, so that at the current iterate $\boldsymbol{w}_t$ we 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 1). To resolve the ambiguity, we could simply replace the gradient $\nabla J(\boldsymbol{w}_t)$ in (1) with some 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

Algorithm 1 Subgradient BFGS (subBFGS)

Initialize: $t := 0, \boldsymbol{w}_0 = \boldsymbol{0}, \boldsymbol{B}_0 = \boldsymbol{I}$;
Set direction-finding stopping tolerances $\epsilon, k_{\max} \in \mathbb{R}_+$;
Compute subgradient $\boldsymbol{g}_0 \in \partial J(\boldsymbol{w}_0)$;
while not converged **do**
  $\boldsymbol{p}_t =$ `descentDirection`$(\boldsymbol{g}_t, \epsilon, k_{\max})$; (Algorithm 2)
  if $\boldsymbol{p}_t =$ failure **then**
    Return $\boldsymbol{w}_t$;
  end if
  Find η_t that obeys (14); (*e.g.*, Algorithm 3)
  $\boldsymbol{s}_t = \eta_t \boldsymbol{p}_t$;
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{s}_t$;
  Compute subgradient $\boldsymbol{g}_{t+1} \in \partial J(\boldsymbol{w}_{t+1})$;
  $\boldsymbol{y}_t = \boldsymbol{g}_{t+1} - \boldsymbol{g}_t$;
  Update $\boldsymbol{B}_{t+1}$ via (5);
  $t := t + 1$;
end while

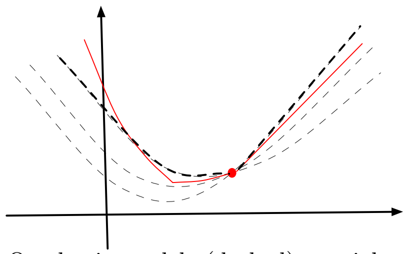

Figure 1. Quadratic models (dashed) *vs.* tightest pseudo-quadratic fit (7) (bold dashes) to the objective function (solid line) at a subdifferentiable point (solid disk).

local quadratic model as follows:

$$\begin{aligned} Q_t(\boldsymbol{p}) &:= J(\boldsymbol{w}_t) + M_t(\boldsymbol{p}), \quad \text{where} \\ M_t(\boldsymbol{p}) &:= \tfrac{1}{2}\boldsymbol{p}^\top \boldsymbol{B}_t^{-1} \boldsymbol{p} \;+\; \sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}. \end{aligned} \tag{7}$$

Note that where J is differentiable, (7) 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 1).

Ideally, we would like to minimize $Q_t(\boldsymbol{p})$, or equivalently $M_t(\boldsymbol{p})$, in (7) to obtain the best search direction,

$$\boldsymbol{p}^* := \operatorname*{arginf}_{\boldsymbol{p} \in \mathbb{R}^d} M_t(\boldsymbol{p}). \tag{8}$$

This is generally intractable due to the presence of a supremum over the entire subdifferential set $\partial J(\boldsymbol{w}_t)$. In many machine learning problems, however, the set $\partial J(\boldsymbol{w}_t)$ has some special structure that simplifies calculation of the supremum in (7). In what follows, we develop an iteration that is guaranteed to find a quasi-Newton descent direction, assuming an oracle that supplies $\operatorname{argsup}_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}$ for a given direction

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

input subgradient $\boldsymbol{g}^{(1)} \in \partial J(\boldsymbol{w}_t)$,
tolerance $\epsilon \in \mathbb{R}_+$, iteration limit $k_{\max}$;
output descent direction $\boldsymbol{p}_t$;
1: Initialize: $i := 1$, $\bar{\boldsymbol{g}}^{(1)} = \boldsymbol{g}^{(1)}$, $\boldsymbol{p}^{(1)} = -\boldsymbol{B}_t \boldsymbol{g}^{(1)}$;
2: $\boldsymbol{g}^{(2)} = \mathrm{argsup}_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}^{(1)}$;
3: Calculate $\epsilon^{(1)}$ via (13);
4: **while** $(\boldsymbol{g}^{(i+1)\top} \boldsymbol{p}^{(i)} > 0$ or $\epsilon^{(i)} > \epsilon)$ and $i < k_{\max}$ **do**
5: $\quad \mu^* := \min \left[1, \dfrac{(\boldsymbol{g}^{(i+1)} - \bar{\boldsymbol{g}}^{(i)})^\top \boldsymbol{p}^{(i)}}{(\boldsymbol{g}^{(i+1)} - \bar{\boldsymbol{g}}^{(i)})^\top \boldsymbol{B}_t (\boldsymbol{g}^{(i+1)} - \bar{\boldsymbol{g}}^{(i)})}\right]$;
6: $\quad \bar{\boldsymbol{g}}^{(i+1)} = (1 - \mu^*) \bar{\boldsymbol{g}}^{(i)} + \mu^* \boldsymbol{g}^{(i+1)}$;
7: $\quad \boldsymbol{p}^{(i+1)} = (1 - \mu^*) \boldsymbol{p}^{(i)} - \mu^* \boldsymbol{B}_t \boldsymbol{g}^{(i+1)}$;
8: $\quad \boldsymbol{g}^{(i+2)} = \mathrm{argsup}_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}^{(i+1)}$;
9: $\quad$ Calculate $\epsilon^{(i+1)}$ via (13);
10: $\quad i := i + 1$;
11: **end while**
12: **if** $\boldsymbol{g}^{(i+1)\top} \boldsymbol{p}^{(i)} > 0$ **then**
13: $\quad$ **return** failure;
14: **else**
15: $\quad$ **return** $\mathrm{argmin}_{j \le i} M_t(\boldsymbol{p}^{(j)})$.
16: **end if**

$\boldsymbol{p} \in \mathbb{R}^d$. In Section 3.1 we provide an efficient implementation of such an oracle for L_2-regularized risk minimization with the hinge loss.

2.2. Finding a Descent Direction

A direction $\boldsymbol{p}_t$ is a descent direction if and only if $\boldsymbol{g}^\top \boldsymbol{p}_t < 0 \ \forall \boldsymbol{g} \in \partial J(\boldsymbol{w}_t)$ (Belloni, 2005), or equivalently

$$\sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}_t \ < \ 0. \tag{9}$$

In particular, 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 (9), making it impossible to find a step that obeys (4), thus causing a failure of the line search. We now present an iterative approach to finding a quasi-Newton *descent* direction.

Inspired by bundle methods (Teo et al., 2007), we build the following convex lower bound on $M_t(\boldsymbol{p})$:

$$M_t^{(i)}(\boldsymbol{p}) \ := \ \tfrac{1}{2} \boldsymbol{p}^\top \boldsymbol{B}_t^{-1} \boldsymbol{p} + \sup_{j \le i} \boldsymbol{g}^{(j)\top} \boldsymbol{p}, \tag{10}$$

where $i, j \in \mathbb{N}$. Given a $\boldsymbol{p}^{(i)} \in \mathbb{R}^d$ the lower bound (10) is successively tightened by computing

$$\boldsymbol{g}^{(i+1)} := \operatorname*{argsup}_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}^{(i)}, \tag{11}$$

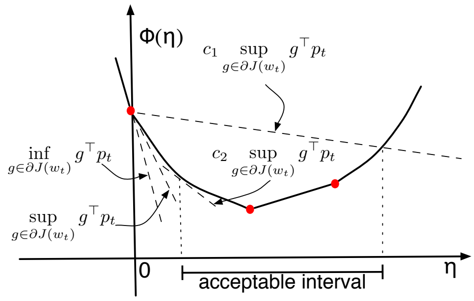

Figure 2. Geometric interpretation of the subgradient Wolfe conditions (14). Solid disks are subdifferentiable points; the slopes of dashed lines are indicated.

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

$$\inf_{\boldsymbol{p}, \xi} \left(\tfrac{1}{2} \boldsymbol{p}^\top \boldsymbol{B}_t^{-1} \boldsymbol{p} + \xi\right) \ \text{s.t.} \ \boldsymbol{g}^{(j)\top} \boldsymbol{p} \le \xi \ \ \forall j \le i. \tag{12}$$

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 $\inf_{\boldsymbol{p} \in \mathbb{R}^d} M_t^{(i)}(\boldsymbol{p})$ 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 $-\boldsymbol{B}_t \boldsymbol{g}^{(i+1)}$. The optimal combination coefficient μ^* can be computed exactly (Step 5 of Algorithm 2) using an argument based on maximizing dual progress. Finally, to derive an implementable stopping criterion, we define $\epsilon^{(i)}$ to be

$$\min_{j \le i} \left[\boldsymbol{p}^{(j)\top} \boldsymbol{g}^{(j+1)} - \tfrac{1}{2}(\boldsymbol{p}^{(j)\top} \bar{\boldsymbol{g}}^{(j)} + \boldsymbol{p}^{(i)\top} \bar{\boldsymbol{g}}^{(i)})\right], \tag{13}$$

where $\bar{\boldsymbol{g}}^{(i)}$ is an aggregated subgradient (Step 6 of Algorithm 2) which lies in the convex hull of $\boldsymbol{g}^{(j)} \in \partial J(\boldsymbol{w}_t) \ \forall j \le i$. $\epsilon^{(i)}$ is monotonically decreasing, and upper bounds the distance from the optimal value of the dual of $M_t(\boldsymbol{p})$, leading us to a practical stopping criterion (Step 4 of Algorithm 2) for our direction-finding procedure. Yu et al. (2008) provide details, and prove that Algorithm 2 converges to the optimal dual objective value with precision ϵ at an $O(1/\epsilon)$ rate.

2.3. 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 $\eta \in \mathbb{R}_+$ which decreases the objective function along the line $\boldsymbol{w}_t + \eta \boldsymbol{p}_t$, *i.e.*, $J(\boldsymbol{w}_t + \eta \boldsymbol{p}_t) =: \Phi(\eta)$. The Wolfe conditions (4) are used in line search routines to enforce a sufficient decrease in the objective value, and to exclude unnecessarily small step sizes (Nocedal and Wright, 1999). However, the original Wolfe conditions require the objective function to be smooth. To extend them to nonsmooth convex problems, we propose the following subgradient reformulation:

$$\begin{aligned} J(\boldsymbol{w}_{t+1}) \;&\leq\; J(\boldsymbol{w}_t) \;+\; c_1\eta_t \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}_t \\ \text{and} \quad \sup_{\boldsymbol{g}'\in\partial J(\boldsymbol{w}_{t+1})} \boldsymbol{g}'^\top \boldsymbol{p}_t \;&\geq\; c_2 \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}_t, \end{aligned} \tag{14}$$

where $0 < c_1 < c_2 < 1$. Figure 2 illustrates how these conditions enforce acceptance of non-trivial step sizes that decrease the objective value. Yu et al. (2008) formally show that for any given descent direction we can always find a positive step size that satisfies (14).

2.4. 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 of $\boldsymbol{B}_t$ with a vector by means of the standard LBFGS matrix-free scheme (Nocedal and Wright, 1999).

3. sub(L)BFGS Implementation for L_2-Regularized Risk Minimization

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

$$J(\boldsymbol{w}) \;:=\; \frac{c}{2}\|\boldsymbol{w}\|^2 \;+\; \frac{1}{n}\sum_{i=1}^{n} l(\boldsymbol{w}^\top\boldsymbol{x}_i, z_i), \tag{15}$$

where $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ are the training instances, $z_i \in \mathcal{Z} \subseteq \mathbb{R}$ 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 $\boldsymbol{w}$ via $\boldsymbol{w}^\top\boldsymbol{x}_i$. A loss function commonly used for binary classification is the hinge loss

$$l(\boldsymbol{w}^\top\boldsymbol{x}, z) \;:=\; \max(0, 1 - z\,\boldsymbol{w}^\top\boldsymbol{x}), \tag{16}$$

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

Differentiating (15) after plugging in (16) yields

$$\partial J(\boldsymbol{w}) = c\,\boldsymbol{w} - \frac{1}{n}\sum_{i=1}^{n}\beta_i z_i \boldsymbol{x}_i = \bar{\boldsymbol{w}} - \frac{1}{n}\sum_{i\in\mathcal{M}}\beta_i z_i \boldsymbol{x}_i, \tag{17}$$

where $\bar{\boldsymbol{w}} := c\,\boldsymbol{w} - \frac{1}{n}\sum_{i\in\mathcal{E}} z_i \boldsymbol{x}_i$ and

$$\beta_i := \begin{cases} 1 & \text{if } i\in\mathcal{E}, \quad \mathcal{E} := \{i : 1 - z_i\boldsymbol{w}^\top\boldsymbol{x}_i > 0\}, \\ [0,1] & \text{if } i\in\mathcal{M}, \quad \mathcal{M} := \{i : 1 - z_i\boldsymbol{w}^\top\boldsymbol{x}_i = 0\}, \\ 0 & \text{if } i\in\mathcal{W}, \quad \mathcal{W} := \{i : 1 - z_i\boldsymbol{w}^\top\boldsymbol{x}_i < 0\}. \end{cases}$$

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

3.1. Realizing the Direction-Finding Method

Recall that our sub(L)BFGS algorithm requires an oracle that provides $\operatorname{argsup}_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top\boldsymbol{p}$ for a given direction $\boldsymbol{p}$. For L_2-regularized risk minimization with binary hinge loss we can implement such an oracle at computational cost linear in the number $|\mathcal{M}_t|$ of current marginal points. (Normally $|\mathcal{M}_t| \ll n$.) Towards this end we use (17) to obtain

$$\begin{aligned} \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top\boldsymbol{p} \;&=\; \sup_{\beta_i, i\in\mathcal{M}_t} \Big(\bar{\boldsymbol{w}}_t - \frac{1}{n}\sum_{i\in\mathcal{M}_t}\beta_i z_i \boldsymbol{x}_i\Big)^{\!\top} \boldsymbol{p} \\ &=\; \bar{\boldsymbol{w}}_t^\top\boldsymbol{p} - \frac{1}{n}\sum_{i\in\mathcal{M}_t}\inf_{\beta_i\in[0,1]} \beta_i z_i \boldsymbol{x}_i^\top\boldsymbol{p}. \end{aligned} \tag{18}$$

Since for a given $\boldsymbol{p}$ the first term of the right-hand side of (18) is a constant, the supremum is attained when we set $\beta_i\ \forall i\in\mathcal{M}_t$ via the following strategy:

$$\beta_i := \begin{cases} 0 & \text{if } \; z_i\boldsymbol{x}_i^\top\boldsymbol{p}_t \;\geq\; 0, \\ 1 & \text{if } \; z_i\boldsymbol{x}_i^\top\boldsymbol{p}_t \;<\; 0. \end{cases} \tag{19}$$

3.2. Implementing the Line Search

The one-dimensional convex function Φ obtained by restricting (15) to a line can be evaluated efficiently. To see this, rewrite the objective as

$$J(\boldsymbol{w}) \;:=\; \frac{c}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\mathbf{1}^\top \max(\mathbf{0}, \mathbf{1} - \boldsymbol{z}\cdot\boldsymbol{X}\boldsymbol{w}), \tag{20}$$

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 $\boldsymbol{p}_t$ at an iterate $\boldsymbol{w}_t$, this allows us to write

$$\begin{aligned} \Phi(\eta) := J(\boldsymbol{w}_t + \eta\boldsymbol{p}_t) = \frac{1}{n}\boldsymbol{\delta}(\eta)^\top[\mathbf{1} - (\boldsymbol{f} + \eta\Delta\boldsymbol{f})] \\ + \frac{c}{2}\|\boldsymbol{w}_t\|^2 + c\eta\boldsymbol{w}_t^\top\boldsymbol{p}_t + \frac{c\eta^2}{2}\|\boldsymbol{p}_t\|^2 \end{aligned} \tag{21}$$

where $\boldsymbol{f} := \boldsymbol{z}\cdot\boldsymbol{X}\boldsymbol{w}_t$, $\Delta\boldsymbol{f} := \boldsymbol{z}\cdot\boldsymbol{X}\boldsymbol{p}_t$, and

$$\delta_i(\eta) := \begin{cases} 1 & \text{if } \; f_i + \eta\,\Delta f_i \;<\; 1, \\ [0,1] & \text{if } \; f_i + \eta\,\Delta f_i \;=\; 1, \\ 0 & \text{if } \; f_i + \eta\,\Delta f_i \;>\; 1 \end{cases} \tag{22}$$

for $1 \leq i \leq n$. We cache $\boldsymbol{f}$ and $\Delta\boldsymbol{f}$, expending $O(nd)$ computational effort. We also cache $\frac{c}{2}\|\boldsymbol{w}_t\|^2$, $c\boldsymbol{w}_t^\top\boldsymbol{p}_t$, and $\frac{c}{2}\|\boldsymbol{p}_t\|^2$, each of which requires $O(n)$ work. The

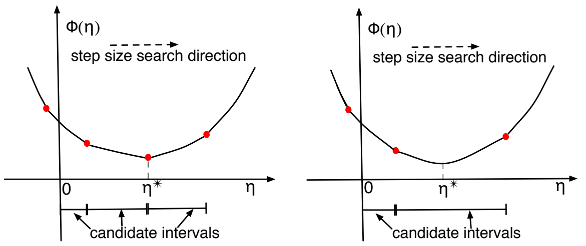

Figure 3. Nonsmooth convex function Φ of step size η. Solid disks are subdifferentiable points; the optimal η^* falls on such a point (left), or between two such points (right).

evaluation of $\boldsymbol{\delta}(\eta)$ and its inner product with $\mathbf{1} - (\boldsymbol{f} + \eta\,\Delta\boldsymbol{f})$ both take $O(n)$ effort. All other terms in (21) can be computed in constant time, thus reducing the amortized cost of evaluating $\Phi(\eta)$ to $O(n)$. We are now in a position to introduce an exact line search which takes advantage of this scheme.

3.2.1. Exact Line Search

Differentiating (21) with respect to η and setting the gradient to zero shows that $\eta^* := \mathrm{argmin}_\eta\, \Phi(\eta)$ satisfies $\eta^* = (\boldsymbol{\delta}(\eta^*)^\top \Delta\boldsymbol{f}/n - c\,\boldsymbol{w}_t^\top \boldsymbol{p}_t)/(c\,\|\boldsymbol{p}_t\|^2)$. It is easy to verify that $\Phi(\eta)$ is piecewise quadratic, and differentiable everywhere except at $\eta_i := (1-f_i)/\Delta f_i$, where it becomes subdifferentiable. At these points an element of the indicator function $\boldsymbol{\delta}(\eta)$ (22) changes from 0 to 1 or vice versa; otherwise $\boldsymbol{\delta}(\eta)$ remains constant. Thus for a smooth interval (η_a, η_b) between subdifferentiable points η_a and η_b (*cf.* Figure 3), if the candidate step

$$\eta^*_{a,b} = \frac{\boldsymbol{\delta}(\eta')^\top \Delta\boldsymbol{f}/n - c\,\boldsymbol{w}_t^\top \boldsymbol{p}_t}{c\,\|\boldsymbol{p}_t\|^2}, \quad \eta' \in (\eta_a, \eta_b) \qquad (23)$$

lies in the interval, it is optimal: $\eta^*_{a,b} \in [\eta_a, \eta_b] \Rightarrow \eta^* = \eta^*_{a,b}$. Otherwise, the interval boundary is optimal if its subdifferential contains zero: $0 \in \partial\,\Phi(\eta_a) \Rightarrow \eta^* = \eta_a$. Sorting the subdifferentiable points η_i facilitates efficient search over intervals; see Algorithm 3 for details.

Algorithm 3 $\eta = \texttt{linesearch}(\boldsymbol{w}_t, \boldsymbol{p}_t, c, \boldsymbol{f}, \Delta\boldsymbol{f})$

input $\boldsymbol{w}_t, \boldsymbol{p}_t, c, \boldsymbol{f}$, and $\Delta\boldsymbol{f}$ as in (21);
output step size η;
$b = c\,\boldsymbol{w}_t^\top \boldsymbol{p}_t,\;\; h = c\,\|\boldsymbol{p}_t\|^2$;
$n = \texttt{length}(\boldsymbol{f}),\;\; j := 1$;
$\boldsymbol{\alpha} := [(\mathbf{1} - \boldsymbol{f})/\Delta\boldsymbol{f}, 0]$; (subdifferentiable points)
$\boldsymbol{\pi} = \texttt{sort}(\boldsymbol{\alpha})$; (index vector)
while $\alpha_{\pi_j} \le 0$ **do**
  $j := j + 1$;
end while
$\eta := \alpha_{\pi_j}/2$;
for $i := 1$ to n **do**
  $\delta_i := \begin{cases} 1 & \text{if } f_i + \eta\,\Delta f_i < 1; \\ 0 & \text{otherwise}; \end{cases}$
end for
$\varrho := \boldsymbol{\delta}^\top \Delta\boldsymbol{f}/n$;
while $j \le \texttt{length}(\boldsymbol{\pi})$ **do**
  $\eta := (\varrho - b)/h$; (candidate step)
  if $\eta \in [\alpha_{\pi_{j-1}}, \alpha_{\pi_j}]$ **then**
    return η;
  else if $\eta < \alpha_{\pi_{j-1}}$ **then**
    return $\eta := \alpha_{\pi_{j-1}}$;
  else
    repeat
      $\varrho := \begin{cases} \varrho - \Delta f_{\pi_j}/n & \text{if } \delta_{\pi_j} = 1, \\ \varrho + \Delta f_{\pi_j}/n & \text{otherwise}; \end{cases}$
      $j := j + 1$;
    until $\alpha_{\pi_j} \ne \alpha_{\pi_{j-1}}$
  end if
end while

4. Related Work

Lukšan and Vlček (1999) propose an extension of BFGS to nonsmooth convex problems. Their algorithm samples gradients around non-differentiable points in order to obtain a descent direction. In many machine learning problems evaluating the objective function and its gradient is very expensive. Therefore, our direction-finding algorithm (Algorithm 2) repeatedly samples subgradients from the set $\partial J(\boldsymbol{w})$ via the oracle, which is computationally more efficient.

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

$$J(\boldsymbol{w}) := c\|\boldsymbol{w}\|_1 + \frac{1}{n}\sum_{i=1}^{n} \ln(1 + e^{-z_i \boldsymbol{w}^\top \boldsymbol{x}_i}), \qquad (24)$$

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 the L_2-regularized hinge loss; the direction finding and line search methods that we discussed in Sections 3.1 and 3.2, 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 $\boldsymbol{B}^{\mathrm{ow}}$ 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 $\boldsymbol{g}^{\mathrm{ow}}$ (Andrew and Gao, 2007, Equation 4) to produce the quasi-Newton direction, $\boldsymbol{p}^{\mathrm{ow}} = \pi(\boldsymbol{p}, \boldsymbol{g}^{\mathrm{ow}})$, where $\boldsymbol{p} := -\boldsymbol{B}^{\mathrm{ow}}\boldsymbol{g}^{\mathrm{ow}}$ and the projection π returns a search direction by setting the i^{th} element of $\boldsymbol{p}$ to zero whenever $p_i g_i^{\mathrm{ow}} > 0$. As shown in Section 5, the direction-finding subroutine of OWL-QN can be replaced by Al-

Table 1. Datasets, regularization constants c, direction-finding convergence criterion ϵ, and the overall number k of direction-finding iterations for L_1-regularized logistic loss and L_2-regularized hinge loss minimization tasks, respectively.

Dataset	Tr./Test Data	Dimen.	Density	c_{L_1}	ϵ_{L_1}	k_{L_1}	$k_{L_1\text{rand}}$	c_{L_2}	ϵ_{L_2}	k_{L_2}
Covertype	522911/58101	54	22.22%	10^{-6}	10^{-5}	0	0	10^{-6}	10^{-8}	44
CCAT	781265/23149	47236	0.16%	10^{-6}	10^{-5}	356	467	10^{-4}	10^{-8}	66
Astro	29882/32487	99757	0.077%	10^{-5}	10^{-3}	1668	2840	$5 \cdot 10^{-5}$	10^{-8}	17
MNIST	60000/10000	780	19.22%	10^{-4}	10^{-5}	60	102	$1.4286 \cdot 10^{-6}$	10^{-8}	244

gorithm 2, which in turn makes the algorithm more robust to the choice of subgradients.

Many optimization techniques use past gradients to build a model of the objective function. Bundle method solvers like SVMStruct (Joachims, 2006) and BMRM (Teo et al., 2007) use them 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.

Vojtěch and Sonnenburg (2007) speed up the convergence of a bundle method solver for the L_2-regularized binary hinge loss. Their main idea is to perform a line search along the line connecting two successive iterates of a bundle method solver. Although developed independently, their line search algorithm is very reminiscent of the method we describe in Section 3.2.1.

5. Experiments

We now evaluate the performance of our subLBFGS algorithm, and compare it to other state-of-the-art nonsmooth optimization methods on L_2-regularized hinge loss minimization. We also compare a variant of OWL-QN that uses our direction-finding routine to the original on L_1-regularized logistic loss minimization.

Our experiments used four datasets: the Covertype dataset of Blackard, Jock & Dean, CCAT from the Reuters RCV1 collection, the Astro-physics dataset of abstracts of scientific papers from the Physics ArXiv (Joachims, 2006), and the MNIST dataset of handwritten digits with two classes: even and odd digits. We used subLBFGS with a buffer of size $m = 15$ throughout. Table 1 summarizes our parameter settings, and reports the overall number of direction-finding iterations for all experiments. We followed the choices of Vojtěch and Sonnenburg (2007) for the L_2 regularization constants; for L_1 they were chosen from the set $10^{\{-6,-5,\cdots,-1\}}$ to achieve the lowest test error.

On convex problems such as these every convergent optimizer will reach the same solution; comparing generalisation performance is therefore pointless. We combined training and test datasets to evaluate the convergence of each algorithm in terms of the objective function value *vs.* CPU seconds. All experiments were carried out on a Linux machine with dual 2.8 GHz Xeon processors with 4GB RAM.

5.1. L_2-Regularized Hinge Loss

For our first set of experiments, we applied subLBFGS together with our exact line search (Algorithm 3) to the task of L_2-regularized hinge loss minimization. Our control methods are the bundle method solver BMRM (Teo et al., 2007) and an optimized cutting plane algorithm, OCAS version 0.6.0 (Vojtěch and Sonnenburg, 2007), both of which demonstrated strong results on the L_2-regularized hinge loss minimization in their corresponding papers.

Figure 4 shows that subLBFGS (solid) reaches the neighbourhood of the optimum (less than 10^{-3} away) noticeably (up to 7 times) faster than BMRM (dashed). As BMRM's approximation to the objective function improves over the course of optimization, it gradually catches up with subLBFGS, though ultimately subLBFGS still converges faster on 3 out of 4 datasets. The performance of subLBFGS and OCAS (dash-dotted) are very similar: OCAS converges slightly faster than subLBFGS on the Astrophysics dataset but is outperformed by subLBFGS on the MNIST dataset.

5.2. 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) as an alternative direction-finding method, such that $\boldsymbol{p}^{\text{ow}} = \texttt{descentDirection}(\boldsymbol{g}^{\text{ow}}, \epsilon, k_{\max})$,[1] and compared this variant (denoted by OWL-QN*) with the original on L_1-regularized logistic loss minimization.

Using the stopping criterion suggested by Andrew and

[1] Note that for the objective (24) it is trivial to construct an oracle that supplies $\text{argsup}_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}$.

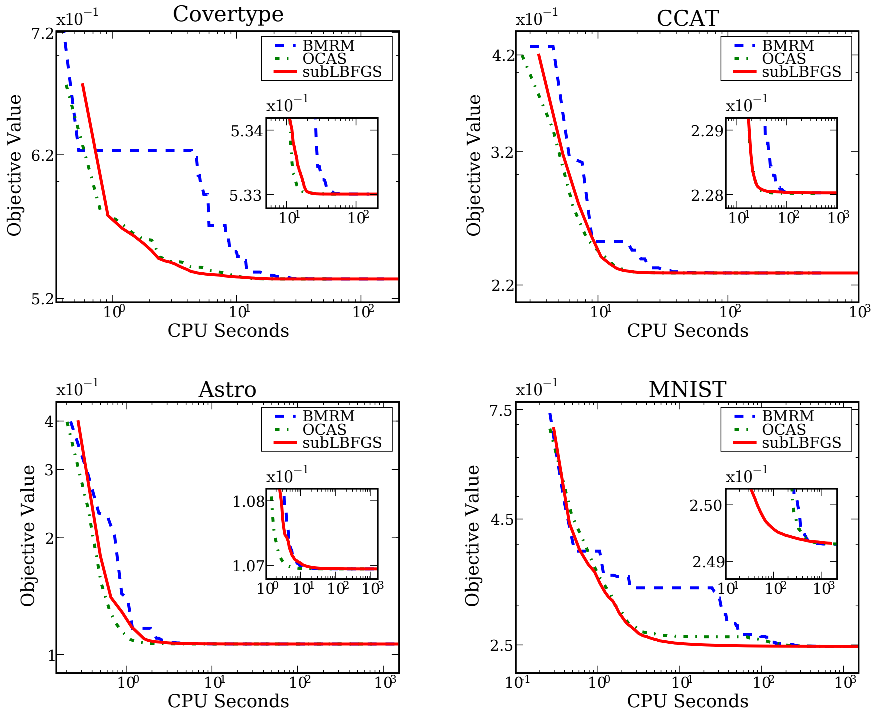

Figure 4. Objective function value *vs.* CPU seconds on L_2-regularized hinge loss minimization tasks.

Gao (2007), we run experiments until the averaged relative change in the objective value over the previous 5 iterations falls below 10^{-5}. Figure 5 shows only minor differences in convergence between the two algorithms.

To examine the algorithms' sensitivity to the choice of subgradients, we also ran them with random subgradients (as opposed to the specially chosen $\boldsymbol{g}^{\text{ow}}$ used before) 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 dataset where we could even plot its (poor) performance was Covertype (dotted OWL-QN(2) line in Figure 5). Our direction-finding routine, by contrast, is self-correcting and thus not affected by this manipulation: The curves for OWL-QN*(2) (plotted for Covertype in Figure 5) lie virtually on top of those for OWL-QN*. Table 1 shows that in this case more direction-finding iterations are needed, *i.e.*, $k_{L_1\text{rand}} \geq k_{L_1}$. This empirically confirms that as long as $\operatorname{argsup}_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}$ is given, Algorithm 2 can indeed be used as a canned quasi-Newton direction-finding routine.

6. Outlook and Discussion

We proposed an extension of BFGS suitable for handling nonsmooth problems often encountered in the machine learning context. As our experiments show, our algorithms are versatile and applicable to many problems, while their performance is comparable to if not better than that of their counterparts in custom-built solvers.

In some experiments we observe that subLBFGS initially makes rapid progress towards the solution but slows down closer to the optimum. We hypothesize that initially its quadratic model allows subLBFGS to make rapid progress, but closer to the optimum it is no longer an accurate model of an objective function dominated by the nonsmooth hinges. We are therefore contemplating hybrid solvers which seamlessly switch between sub(L)BFGS and bundle solvers.

In this paper we applied subLBFGS to L_2-regularized risk minimization with binary hinge loss. It can also be extended to deal with generalizations of the hinge loss, such as multi-class, multi-category, and ordinal regression problems; this is part of our ongoing research.

Finally, to put our contributions in perspective, recall that we modified three aspects of the standard BFGS algorithm, namely the quadratic model (Section 2.1), the descent direction finding (Section 2.2), and the line search (Section 2.3). Each of these modifications is versatile enough to be used as a component in other nonsmooth optimization algorithms. This not only of-

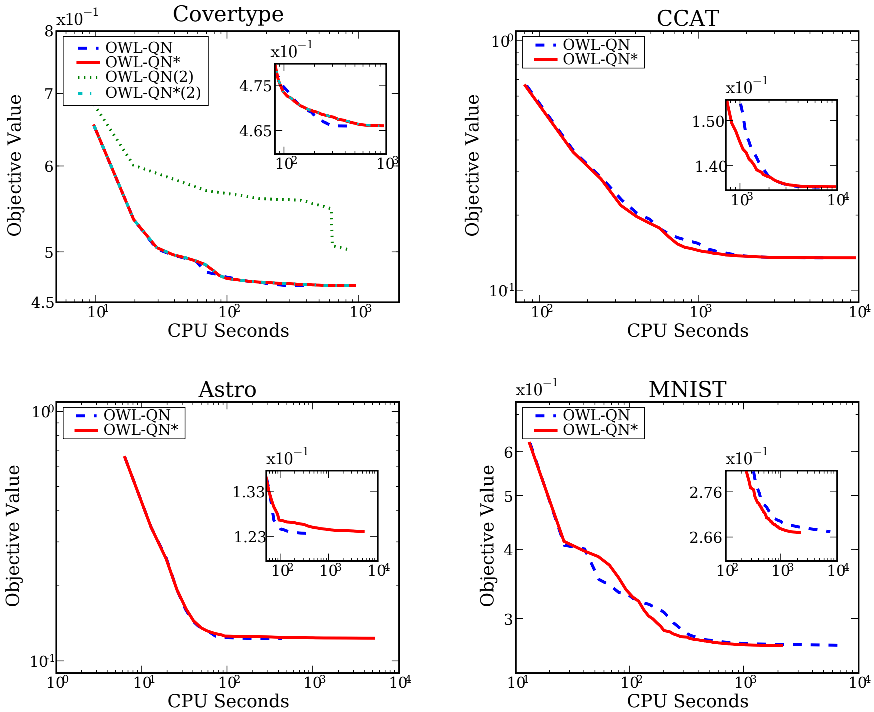

Figure 5. Objective function value *vs.* CPU seconds on L_1-regularized logistic loss minimization tasks.

fers the promise of improving existing algorithms, but may also help clarify connections between them. We hope that this will focus attention on those core subroutines that need to be made more efficient in order to handle larger and larger datasets.

Acknowledgement

NICTA is funded by the Australian Government's Backing Australia's Ability and the Centre of Excellence programs. This work is also supported by the IST Program of the European Community, under the FP7 Network of Excellence, ICT-216886-NOE.

References

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.

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

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

J. 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.

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.

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

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

C. Teo, Q. Le, A. Smola, and S. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD)*. ACM, 2007.

F. Vojtěch and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. Technical Report 1, Fraunhofer Institute FIRST, December 2007. http://publica.fraunhofer.de/documents/N-66225.html.

J. Yu, S. V. N. Vishwanathan, S. Günter, and N. N. Schraudolph. A quasi-Newton approach to nonsmooth convex optimization. Technical Report arXiv:0804.3835, April 2008. http://arxiv.org/pdf/0804.3835.