# SECOND-ORDER METHODS FOR L1 REGULARIZED PROBLEMS IN MACHINE LEARNING

*Samantha Hansen and Jorge Nocedal*

Northwestern University

## ABSTRACT

This paper proposes a Hessian-free Newton method for solving large-scale convex functions with an L1 regularization term. These problems arise in supervised machine learning models in which it is important to seek a sparse parameter vector. The proposed method operates in a batch setting, which is well suited for parallel computing environments, and employs sub-sampled Hessian information to accelerate progress of the iteration. The method consists of two phases, an active-set prediction phase that employs first-order and second-order information, and subspace phase that performs a Newton-like step. Numerical results on a speech recognition problem illustrate the practical behavior of the method.

***Index Terms—*** Logistic Regression, L1 Regularization, Newton Method, Iterative Shrinkage, Hessian-Free Newton.

## 1. INTRODUCTION

Sub-sampled Hessian Newton methods have recently been proposed for machine learning and stochastic optimization applications [1, 2]. The novel feature in these batch methods is the use of a much smaller sample for Hessian computations than for function and gradient evaluations. This *sub-sampling Hessian strategy* overcomes the main obstacle in the application of Newton-like methods to some large-scale machine learning problems, namely the prohibitive cost of forming the Hessian, or computing Hessian-vector products. By choosing a small enough Hessian sub-sample, the cost of the iteration is comparable to the cost of one gradient evaluation, but the resulting algorithm is much faster than a gradient method. Sub-sampled Hessian Newton methods of this kind have been applied successfully to multinomial logistic regression problems [1] and neural networks [2]. In those studies, the underlying objective function is smooth.

In this paper, we study the design of a sub-sampled Hessian Newton method for non-smooth optimization problems of the form

$$J(w) = f(w) + c||w||_1, \tag{1}$$

where $f$ is a smooth convex function and $c$ is a constant that determines the weight of the $L_1$ regularization term. The function $f$ often has the form

$$f(w) = \frac{1}{N}\sum_{i=1}^{N} \ell(h(w; x_i), y_i), \tag{2}$$

where $\ell$ is a loss function, $(x_i, y_i)$ denote the training points, $h$ is a linear function, and $N$ is the size of the training set.

In the last 15 years, many first-order methods have been designed for solving problem $(1)$, using stochastic or batch approaches; see [3, 4] and the references therein. More recently, several methods have been proposed that incorporate second-order information, either as an integral part of the iteration [5] or as a means to refine the solution and yield fast final convergence [6, 7, 8]. In this paper, we propose a method that operates in a batch setting and employs sub-sampled Hessian information *at every iteration*. The design of the method is guided by the observation that sparse solutions can be obtained quickly by a combination of a prediction phase that makes a tentative guess of zero components in the solution vector, and a subspace phase that improves upon this guess through the use of second-order information.

## 2. HESSIAN SUB-SAMPLING

The proposed algorithm employs second-order information based on a small fraction of the training points used in the function and gradient computation. At the beginning of each iteration, the algorithm chooses a set $\mathcal{H}_k \subset \{1, 2, \ldots, N\}$, and defines the Hessian approximation

$$B_k := \frac{1}{|\mathcal{H}_k|}\sum_{i\in\mathcal{H}_k} \nabla^2 \ell(h(w_k; x_i), y_i). \tag{3}$$

Thus, $B_k$ is the Hessian of the batch approximation to $(2)$ based on the sample $\mathcal{H}_k$. The matrix $(3)$ is never formed; instead, the algorithm computes products of $B_k$ times vectors — and these products can be coded directly. An observation that lies at the heart of our approach is that the cost of a multiplication of $B_k$ times a vector increases linearly with the size of $\mathcal{H}_k$. Specifically, the cost of a Hessian-vector product is of order $O(|\mathcal{H}_k|/N)$ times the cost of a gradient evaluation.

## 3. TWO PHASE APPROACH

### 3.1. Overview

The objective function (1) is not smooth, and it is difficult to design a second-order method for minimizing it directly without incurring a high computational cost. To resolve this difficulty, we follow a two-phase approach based on the observation that the function $J : R^n \to R$ is smooth in the relative interior of any given orthant in $R^n$.

The goal of the prediction phase is to identify an orthant $\Omega$ that may contain the solution of the problem. In our algorithm, the prediction phase is based on iterative shrinkage, a method that primarily uses first-order information. The orthant defined by the prediction phase defines the sign of each component of $w$ and also identifies variables that should be kept at zero. In the subspace minimization phase, a quadratic model of $J$ is minimized over the space of the remaining variables. The minimization is performed approximately, over $\Omega$, using a Hessian-free conjugate gradient (CG) method [9]. A distinctive feature of our method is that the Hessian of the quadratic model, which is given by (3), is constructed using a small sample of training points — as small as 5% of the size of the sample used in the computation of the function and gradients. This makes each CG iteration very inexpensive, and makes it practical to perform the subspace minimization phase at every iteration of the algorithm.

### 3.2. Prediction Phase

The prediction step is based on the iterative shrinkage method [10], which at the current iterate $w_k$, generates a point $w_k^c$ as follows:

$$w_k^c = \arg\min_z \tfrac{1}{2}\left\|z - u_k\right\|_2^2 + c\alpha_k \left\|z\right\|_1,$$

where $u_k = w_k - \alpha_k \nabla f(w_k)$ and $\alpha_k$ is a steplength parameter. The component wise solution of this problem is given by

$$[w_k^c]^i = \text{sign}(u_k^i)\max\left(|u_k^i| - c\alpha_k, 0\right). \tag{4}$$

In our method, we define the steplength as

$$\alpha_k = \frac{\nabla f(w_k)^T \nabla f(w_k)}{\nabla f(w_k)^T B_k \nabla f(w_k)}, \tag{5}$$

where $B_k$ is the sub-sampled Hessian defined in (3). This choice of $\alpha_k$ corresponds to the minimizer of a quadratic model of $f$ along the direction $-\nabla f(w_k)$. Our computational experience suggests that (5) is more effective than the Barzilai-Borwein parameter used in [3].

To define the active orthant, it is convenient [5] to introduce the vector

$$z_k^i = \text{sign}([w_k^c]^i). \tag{6}$$

The variables $w^i$ with $z_k^i = 0$ are kept at zero during the subspace minimization phase, while the rest of the variables are free. The active orthant is defined as

$$\Omega_k = \{w \mid \text{sign}(w^i) = \text{sign}(z_k^i)\}. \tag{7}$$

*Modification to the Iterative Shrinkage Step.* We have observed that the prediction made by the iterative shrinkage step is is not always effective. Iterative shrinkage allows nonzero variables to move, but the subspace phase, which uses second-order information, may reset a large fraction of these variables to zero. To avoid the unnecessary projection of variables to zero, we could allow the prediction phase to move a variable only if the sign of the iterative shrinkage step coincides with the sign of a Newton step.

Specifically, at the start of the prediction phase, we compute a step $d_k^N$ step by solving the system

$$B_k d = -\nabla f(w_k),$$

approximately, using the Hessian-free conjugate gradient method [9], where $B_k$ is the sub-sampled Hessian defined in (3). A component $[w_k^c]^i$ such that $[\nabla f(w_k)]^i [d_k^N]^i > 0$ is considered to be potentially unreliable, and our strategy is to reset it to the current iterate $[w_k]^i$. This is equivalent to setting $\alpha_k = 0$ for the unreliable components and letting $\alpha_k$ be defined by (5) for the other components. The modification becomes

$$[w_k^c]^i \leftarrow \begin{cases} [w_k^c]^i & \text{if } [d_k^N]^i \times \nabla f(w_k)^i \le 0 \\ [w_k]^i & \text{otherwise.} \end{cases} \tag{8}$$

This modification need not be applied at every iteration. Our experience indicates that it is most useful in the early stages of the optimization. In order to ensure global convergence, we need to impose the requirement that the modification (8) be applied only a finite number of times.

### 3.3. Subspace Minimization and Line Search

Having determined the prediction point $w_k^c$, and the corresponding orthant $\Omega_k$ through (6), (7), (8), we minimize a quadratic model of the objective function, over the space of free variables. Specifically, we apply the Hessian-free CG algorithm to compute an approximate solution $w_k^s$ of the quadratic problem

$$\begin{aligned} \min_w \ & b_k^T (w - w_k^c) + \tfrac{1}{2}(w - w_k^c)^T B_k (w^c - w_k) \\ \text{s.t. } & w_i = 0, \text{ for all } i \text{ such that } z_k^i = 0 \end{aligned}$$

where

$$b_k = g_k + B_k(w_k^c - w_k), \qquad g_k = \nabla f(w_k) + c z_k, \tag{9}$$

and $z_k$ is defined in (6). The choice (3) for $B_k$ is motivated by the fact that in the interior of any orthant in $R^n$, the Hessian of $J$ is equal to $\nabla^2 f$. As noted above, $B_k$ is never formed

explicitly, but its product with vectors is coded directly; see e.g. [1] for an example of this computation in a multinomial logistic regression problem.

To ensure decrease in the objective function, the algorithm performs a backtracking line search along a piecewise linear path that starts at $w_k^s$ and ends at $w_k$. Specifically, the algorithm first tries to compute a step length $\hat{\alpha}_k \in (\alpha_{min}, 1]$ that satisfies the sufficient decrease condition

$$J(P(w_k^c + \hat{\alpha}_k d_k)) \leq J(w_k) + \sigma g_k^T (P[w_k^c + \hat{\alpha}_k d_k] - w_k), \tag{10}$$

where $\sigma = 10^{-4}$, $g_k$ is defined in (9), $d_k = w_k^s - w_k^c$, and $P(\cdot)$ denotes the orthogonal projection onto the orthant $\Omega_k$, i.e.,

$$P(w^i) = \begin{cases} w^i & \text{if sign}(w^i) = \text{sign}(z_k^i) \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

If this search is unsuccessful, the line search moves to the iterative shrinkage path $s(\tilde{\alpha})$ given by

$$s^i(\tilde{\alpha}) \equiv \text{sign}(u_k^i(\tilde{\alpha})) \max\left(|u_k^i(\tilde{\alpha})| - c\tilde{\alpha}, 0\right), \quad \tilde{\alpha} \in (0, \bar{\alpha}_k^0],$$

where $u_k(\tilde{\alpha}) = w_k - \tilde{\alpha}\nabla f(w_k)$ and $\bar{\alpha}_k^0$ is given by the right hand side in (5). By backtracking along this path, the line search finds a steplength $\tilde{\alpha}_k$ that yields sufficient decrease in $J$.

In summary, the new iterate of the algorithm is defined as

$$w_{k+1} = \begin{cases} P[w_k^c + \hat{\alpha}_k d_k] & \hat{\alpha}_k > \alpha_{min} \\ s(\tilde{\alpha}_k) & \text{otherwise.} \end{cases}$$

We refer to the method just outlined as Algorithm I.

## 4. RELATED WORK

The orthant wise limited memory BFGS method (OWL) [5] has some similarities with the method proposed in this paper. The identification phase in OWL can be seen as an infinitesimal line search along the steepest descent direction of the non-differentiable function $J$, at the current iterate $w_k$. After the active orthant has been identified, OWL computes a (projected) limited memory BFGS step in the space of the free variables. OWL has proved to be effective in various $L_1$ regularized problems in machine learning, but its convergence properties are unknown to us (as explained in [11] the convergence proof given in [5] is not correct).

The two-phase methods described in [6, 12] employ second-order information in the subspace phase, but differ in significant ways from the method described here. For example, the method in [6], which is closer in spirit to ours, does not perform a sub-space phase at every iteration, employs an interior point method in the subspace phase, which prevents this phase from forcing components of the solution to zero early on, and does not employ a sub-sampling approach to reduce the cost of the CG step.

The convergence properties of the method proposed in this paper will be analyzed in a future paper; they follow from the results given in [1, 6, 13]. A more interesting question is to establish a complexity bound on the total computational effort of the algorithm, when implemented in a dynamic setting where the sample size used for function and gradient evaluations increases automatically during the progression of the optimization; see [11].

## 5. NUMERICAL TESTS

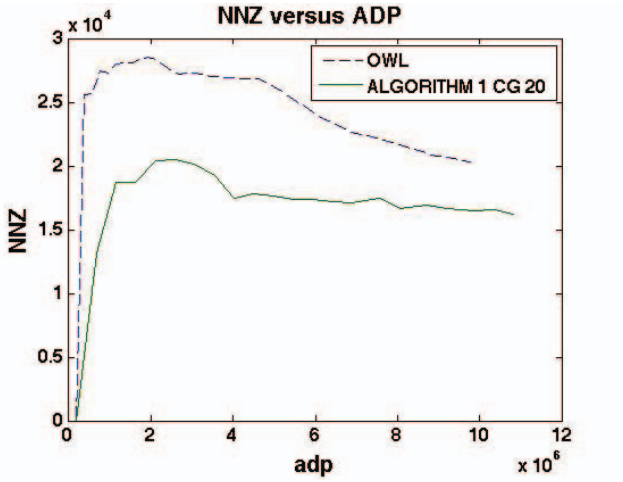


Figure 1: Number of nonzeros vs CPU time

We compare the performance of the sub-sampled Hessian $L_1$ Newton Method (Algorithm I) and the OWL method on a larger version of the problem described in [1], involving multi-class classification of speech frames. We note that sparsity in the final solution normally yields lower generalization error. The training set, which was provided by Google, has $N = 191607$ training points and $n = 30315$ parameters in $w$. The test problem was designed to be small enough to be run on a multi-core workstation, but sufficiently complex to be representative of production-scale speech recognition problems.

For the OWL method we set the memory size to 20 (using a memory of 5 gave similar results), and employed the implementation provided in LibLBFGS.

For Algorithm I, we imposed a limit of 20 CG iterations in the subspace phase, and set the Hessian sub-sampling to be 5%, i.e., $|\mathcal{H}_k|/N = 0.05$. Although the sample size $|\mathcal{H}_k|$ is constant throughout the progression of the algorithm, the sample $\mathcal{H}_k$ itself must be chosen afresh at every (outer) iteration of Algorithm I.

The initial point is given by $w_0 = 0$ in both methods. Figure 1 plots the number of non-zeros in the solution $w_k$, and Figure 2 the probability of correct classification, defined as $\exp(-J(w_k))$, as a function of the number of accessed data points. The latter is a measure of cpu time; it considers

the cost of the function, gradient and Hessian-vector products.

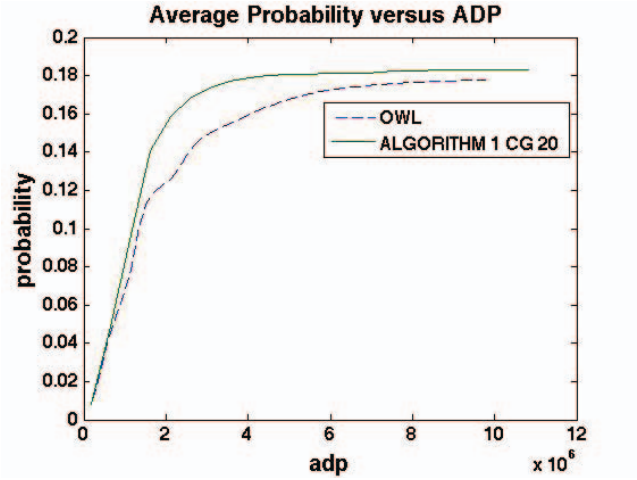


Figure 2: $\exp(-J(w_k))$ vs CPU time

We observe that Algorithm I is more effective than OWL in this test. The greatest benefit is seen in the generation of a sparse solution early on in the iteration.

Controlled tests indicate that the subspace phase plays a crucial role both in the speed of Algorithm I and in its ability to generate a sparse solution quickly. Figure 3 illustrates the behavior of Algorithm I as the number of CG iterations is increased from 5 to 20. Note that performance improves with the number of CG iterations, which corresponds to using Hessian information more thoroughly. (Increasing the number of CG iterations beyond 20 is not beneficial in this problem; the reduction in iterations does not overcome the increased cost of the Newton step computation.)

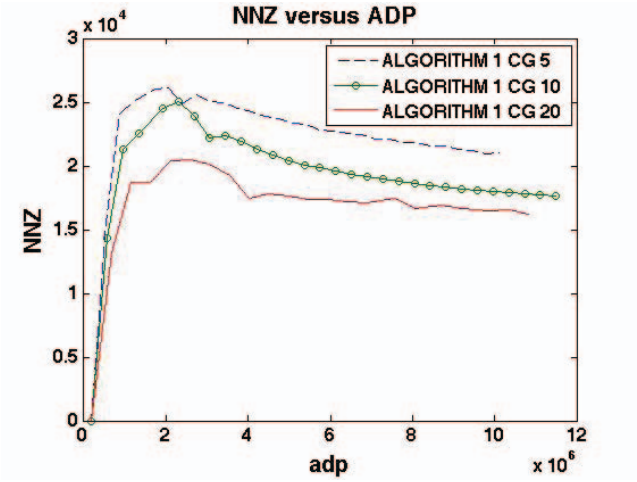


Figure 3: Number of nonzeros vs CPU time

## 6. CONCLUSION

We described a second-order batch method for machine learning models that include an $L_1$ regularization term. The novelty of the approach lies in the use of sub-sampled Hessian information in a subspace minimization that plays a prominent role in the algorithm. Numerical tests on a speech recognition problem show that the use of second-order information accelerates convergence and promotes the fast generation of sparse solutions.

*Acknowledgments.* The authors thank Yuchen Wu and Gillian Chin for their assistance in the development of the code, and Will Neveitt for many useful discussions.

## 7. REFERENCES

[1] Byrd, R., G. M Chin, W. Neveitt, and J. Nocedal, "On the use of stochastic Hessian information in unconstrained optimization," *SIAM Journal on Optimization*, 2011.

[2] J. Martens, "Deep learning via Hessian-free optimization," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.

[3] S.J. Wright, R.D. Nowak, and M.A.T. Figueiredo, "Sparse reconstruction by separable approximation," *Signal Processing, IEEE Transactions on*, vol. 57, no. 7, pp. 2479–2493, 2009.

[4] L. Xiao, "Dual averaging methods for regularized stochastic learning and online optimization," *The Journal of Machine Learning Research*, vol. 9999, pp. 2543–2596, 2010.

[5] G. Andrew and J. Gao, "Scalable training of l 1-regularized log-linear models," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 33–40.

[6] J. Shi, W. Yin, S. Osher, and P. Sajda, "A fast hybrid algorithm for large-scale l 1-regularized logistic regression," *The Journal of Machine Learning Research*, vol. 11, pp. 713–741, 2010.

[7] S.J. Wright, "Accelerated block-coordinate relaxation for regularized optimization," Tech. Rep., Computer Science Department, University of Wisconsin, 2010.

[8] W. Shi, G. Wahba, S. Wright, K. Lee, R. Klein, and B. Klein, "Lasso-patternsearch algorithm with application to ophthalmology and genomic data," *Statistics and its Interface*, vol. 1, no. 1, pp. 137, 2008.

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

[10] D.L. Donoho, "De-noising by soft-thresholding," *Information Theory, IEEE Transactions on*, vol. 41, no. 3, pp. 613–627, 1995.

[11] Byrd, R., G. M Chin, J. Nocedal and Y. Wu, "Sample size selection in optimization methods for machine learning," Tech. Rep., Optimization Center Report 2011/3, Northwestern University, 2011.

[12] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang, "A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization and continuation," *SIAM Journal on Scientific Computing*, vol. 32, no. 4, pp. 1832–1857, 2010.

[13] D. P. Bertsekas, "Projected Newton methods for optimization problems with simple constraints," *SIAM Journal on Control and Optimization*, vol. 20, no. 2, pp. 221–246, 1982.