Subsampled Hessian Newton Methods for Supervised Learning

Chien-Chih Wang, Chun-Heng Huang, Chih-Jen Lin

Department of Computer Science, National Taiwan University, Taipei 10617, Taiwan

Keywords: Subsampled Hessian, Deep Neural Networks, Large-scale classification.

Abstract

Newton methods can be applied in many supervised learning approaches. However, for large-scale data, the use of the whole Hessian matrix can be time consuming. Recently, subsampled Newton methods have been proposed to reduce the computational time by using only a subset of data for calculating an approximation of the Hessian matrix. Unfortunately, we find that in some situations the running speed is worse than the standard Newton method because cheaper but less accurate search directions are used. In this work, we propose some novel techniques to improve the existing subsampled Hessian

Newton method. The main idea is to solve a 2-dimensional sub-problem per iteration to adjust the search direction to better minimize the second-order approximation of the function value. We prove the theoretical convergence of the proposed method. Experiments on logistic regression, linear SVM, maximum entropy, and deep networks indicate that our techniques significantly reduce the running time of the subsampled Hessian Newton method. The resulting algorithm becomes a compelling alternative to the standard Newton method for large-scale data classification.

1 Introduction

The problems we consider arise from supervised learning, which aims to train a model based on observed labeled training data and predict the labels of previously unseen data with the model. Given a set of training examples $(y_i, \boldsymbol{x}_i), i = 1, ..., l$, where y_i is a label (class) and $\boldsymbol{x}_i \in R^n$ is a feature vector, and a loss function ξ parametrized by a weight vector $\boldsymbol{w} \in R^n$, the goal is to minimize the average loss of training data.

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}), \quad \text{where } f(\boldsymbol{w}) \equiv \frac{1}{l} \sum_{i=1}^{l} \xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i). \tag{1}$$

We require that the loss function is convex. For large-scale problems, the leading obstacle is that it can be very costly to evaluate the objective function $f(\boldsymbol{w})$, the gradient $\nabla f(\boldsymbol{w})$ and the Hessian $H = \nabla^2 f(\boldsymbol{w})$ when all the training examples are used. To overcome this obstacle, a sampling framework has been recently proposed by Byrd et al. [2011]. Because training examples are theoretically drawn from some probability distribution, $f(\boldsymbol{w})$ of (1) can be seen as an expected loss. Based on the fact that training examples are often redundant to some extent, we could employ a subset of

training examples instead of the whole training data set to the optimization process. Let $R \subset \{1, \ldots, l\}$ be a training subset. The stochastic approximation of the objective function is defined as

$$f_R(\boldsymbol{w}) \equiv \frac{1}{|R|} \sum_{i \in R} \xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i). \tag{2}$$

To avoid expensive Hessian calculation, Byrd et al. [2011] further select a subset $S \subset R$ to define the following "subsampled Hessian:"

$$H_S \equiv \frac{1}{|S|} \sum_{i \in S} \nabla^2 \xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i). \tag{3}$$

The work by Byrd et al. [2011] then devises a subsampled Hessian Newton method. For a multinomial logistic regression problem, the running speed is shown to be better than that of using the full Hessian. Although Byrd et al. [2011] consider a general setting of having both subsets R and S, in their analysis and experiments, $R = \{1, \ldots, l\}$ is assumed. We follow the same setting and focus on studying the use of subsampled Hessian with S.

In machine learning practice, to avoid overfitting, a regularization term is often considered together with the training loss. Then the objective function becomes

$$f(\boldsymbol{w}) = \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w} + \frac{C}{l} \sum_{i=1}^{l} \xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i),$$

where C is a given regularization parameter. With this modification, $f_R(\boldsymbol{w})$ is no longer the average of subsampled training losses. Therefore, in our analysis we do not restrict $f_R(\boldsymbol{w})$ and H_S to be the same as (2) and (3), respectively. Instead, we consider a more general setting by assuming that for the function to be minimized, its H_S and the full Hessian are related:

$$\bar{m} H_S \preceq H \preceq \bar{M} H_S, \forall \boldsymbol{w}_k, \tag{4}$$

where $\{\boldsymbol{w}_k\}$ is the sequence of iterates generated by the optimization procedure, $\bar{M} \geq \bar{m} \geq 0$ are two constants, and $A \preceq B$ means that $B - A$ is positive semi-definite. For subsampled Hessian Newton methods considered in this paper, we show in Section 5 that (4) easily holds for popular methods like logistic regression, SVM, and maximum entropy.

Some studies other than Byrd et al. [2011] have considered subsampled Hessian Newton methods. For example, Martens [2010] proposed and applied a subsampled Hessian method for training neural networks. Chapelle and Erhan [2011] make an extension to use preconditioned conjugate gradient methods for obtaining search directions.

In this work, we begin with pointing out in Section 2 that for some classification problems, the subsampled Hessian Newton method may be slower than the full Hessian Newton method. The main reason is that, by using only a subset S, the resulting search direction and step size are very different from the full Newton direction that minimizes the second-order approximation of the function reduction. Based on this observation, in Section 3, we propose some novel techniques to improve the subsampled Hessian Newton method. The main idea is to solve a 2-dimensional sub-problem for adjusting the search direction so that the second-order approximation of the function value is better minimized. The theoretical convergence of the proposed methods is given in Section 4. In Section 5, we apply the proposed methods to several machine learning problems: logistic regression (LR), l2-loss support vector machines (SVM), maximum entropy (ME) and deep neural networks.

Our implementation for LR, SVM, and ME is extend from the software LIBLINEAR

[Fan et al., 2008], while for deep networks we extend the implementation in Martens [2010]. Experiments in Section 6 show that the proposed methods are faster than the subsampled Newton method originally proposed in Byrd et al. [2011]. Therefore, our improved subsampled Hessian Newton method can effectively train large-scale data. A supplementary file including additional analysis and experiments is available at `http://www.csie.ntu.edu.tw/~cjlin/papers/sub_hessian/supplement.pdf`.

2 Subsampled Hessian Newton-CG Method and Its Practical Performance

We briefly review the method in Byrd et al. [2011]. At the kth iteration, from the current solution $\boldsymbol{w}_k$ and a given subset S_k, we find a direction $\boldsymbol{d}_k$ by solving the following linear system

$$H_{S_k}\boldsymbol{d}_k = -\nabla f(\boldsymbol{w}_k), \tag{5}$$

where H_{S_k} is the subsampled Hessian defined in (3). The main difference from the standard Newton method is that H_{S_k} rather than the full Hessian $H_k \equiv \nabla^2 f(\boldsymbol{w}_k)$ is used.

For large-scale problems, existing Newton methods often approximately solve the linear system by the conjugate gradient (CG) method, which mainly computes a sequence of Hessian-vector products. Byrd et al. [2011] adopt the same strategy, so subsampled Hessian-vector products are computed. They terminate the CG procedure after either a pre-specified maximal number of CG iterations has been reached or the follow-

Algorithm 1 Subsampled Hessian Newton-CG method

1: Given initial $\boldsymbol{w}_0 = \mathbf{0}$, $\text{CG}_{\max}$ and constants $\eta, \sigma \in (0, 1)$.

2: **for** $k = 0, 1, \ldots$ **do**

3: Choose a subset S_k.

4: Approximately solve (5) by CG to obtain a direction $\boldsymbol{d}_k$ after the condition (6) is satisfied or the number of CG iterations reaches $\text{CG}_{\max}$.

5: Find α_k satisfying (7) by backtracking line search with initial $\alpha = 1$.

6: Update $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \alpha_k \boldsymbol{d}_k$.

7: **end for**

ing inequality has been satisfied.

$$||H_{S_k}\boldsymbol{d}_k + \nabla f(\boldsymbol{w}_k)|| \leq \sigma ||\nabla f(\boldsymbol{w}_k)||, \tag{6}$$

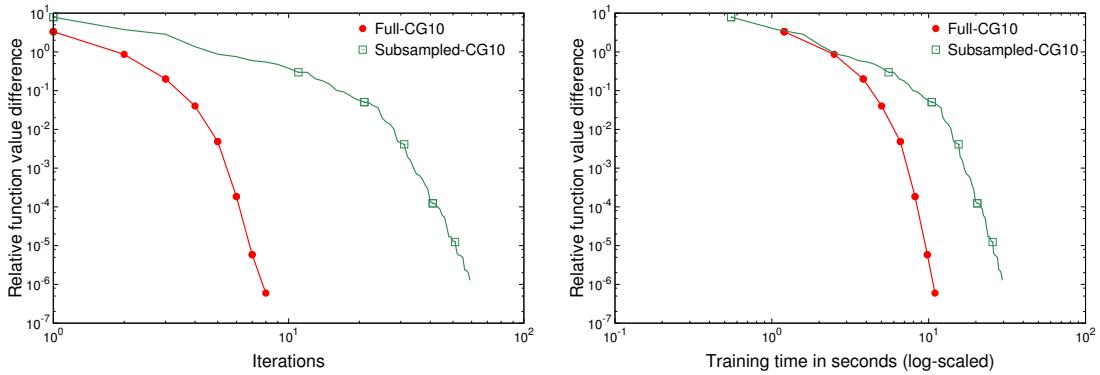
where $\sigma \in (0, 1)$ is a given tolerance. For many machine learning problems, the Hessian (or subsampled Hessian) takes a special form, and hence, the Hessian-vector product in the CG procedure can be conducted without explicitly forming the Hessian. This type of Hessian-free Newton-CG methods for machine learning applications has been used in, for example, Keerthi and DeCoste [2005], Lin et al. [2008].

After obtaining the direction $\boldsymbol{d}_k$, to ensure the convergence, a line search is used to find a step size α_k that satisfies the following sufficient decrease condition

$$f(\boldsymbol{w}_k + \alpha_k \boldsymbol{d}_k) \leq f(\boldsymbol{w}_k) + \eta \alpha_k \nabla f(\boldsymbol{w}_k)^T \boldsymbol{d}_k, \tag{7}$$

where $\eta \in (0, 1)$ is a pre-specified constant. Note that the direction $\boldsymbol{d}_k$ obtained after CG iterations is a descent direction (i.e., $\nabla f(\boldsymbol{w}_k)^T \boldsymbol{d}_k < 0$) ,[1] so (7) is satisfied for some

[1] See references of CG in, for example, Golub and Van Loan [1996].



(a) The closeness to the optimal function value (y-axis) versus iterations (x-axis)

(b) The closeness to the optimal function value (y-axis) versus training time (x-axis)

Figure 1: A comparison between using 5% subsampled Hessian and full Hessian. Data set news20 and logistic regression (LR) are considered. See Section 6 for details of experimental settings. The "CG10" in legends indicates the $\text{CG}_{\max}$ value in Algorithm 1.

α_k. Byrd et al. [2011] suggest using backtracking line search by sequentially trying

$$\alpha_k = 1, \frac{1}{2}, \frac{1}{4}, \dots \tag{8}$$

until (7) holds. Then $\alpha_k \boldsymbol{d}_k$ is used to update $\boldsymbol{w}_k$ to $\boldsymbol{w}_{k+1}$. The overall procedure is summarized in Algorithm 1. It is proved in Byrd et al. [2011] that if the subsampled Hessian satisfies some minor conditions, then Algorithm 1 leads to the convergence of $\{\nabla f(\boldsymbol{w}^k)\}$ to zero. In their experiments, the maximal number of CG iterations is set to be

$$\text{CG}_{\max} = 10.$$

Although solving (5) with subsampled Hessian is cheaper than full Hessian, the less accurate direction may result in slower convergence (i.e., more iterations in the Newton method). In Figure 1, we conduct a simple comparison between using subsampled and

full Hessian. For fairness, all other implementation details are kept the same. We check the relationship between the closeness to the optimal objective value and the following measures.

(a) Number of iterations.

(b) Training time.

It can be clearly seen in Figure 1(a) that the implementation of using subsampled Hessian needs significantly more iterations to converge. We then check running time in Figure 1(b). The difference between the two settings becomes smaller because each iteration of using subsampled Hessian is cheaper. However, the approach of using subsampled Hessian is still worse. Although from Byrd et al. [2011] and our subsequent experiments, subsampled Newton is shown to be faster for some other problems, our example here demonstrates that the opposite result may occur.

The slower convergence of the subsampled Hessian method in Figure 1(a) indicates that its direction is not as good as the full Newton direction. This situation is expected because the sampled set S_k may not represent the full set well. In Section 6, we will see that as the size of S_k shrinks, the performance gets worse.

Based on the above discussion, we aim at improving the subsampled-Hessian method so that it is generally faster than the full-Hessian setting. It is well known that the standard Newton method of using full Hessian solves the following second-order approximation of $f(\boldsymbol{w}_k + \boldsymbol{d}) - f(\boldsymbol{w}_k)$.

$$\min_{\boldsymbol{d}} \ \nabla f(\boldsymbol{w}_k)^T \boldsymbol{d} + \frac{1}{2} \boldsymbol{d}^T H_k \boldsymbol{d}. \tag{9}$$

When $\boldsymbol{w}_k$ is near the optimum, $f(\boldsymbol{w}_k + \boldsymbol{d}) - f(\boldsymbol{w}_k)$ is almost the same as the ap-

proximation in (9). Therefore, not only does the full Newton direction (i.e., $\alpha_k = 1$) satisfy the sufficient decrease condition in (7), but also the method enjoys fast quadratic convergence. We observe in the above experiment that when subsampled Hessian is used, $\alpha_k = 1$ satisfies the condition (7) at a much later stage. Therefore, a crucial reason for the inferior running speed is that the obtained directions are not as good as the full Newton directions. In Section 3, we will propose novel techniques to improve the subsampled Hessian Newton method.

3 Modified Subsampled-Hessian Newton Directions

The main objective of this section is to adjust a subsampled Newton direction so that it gives a smaller objective function value of (9).

To make the direction closer to the full Newton, extra computational operations or data accesses are needed. Let us check what we may be able to afford. At each iteration of a standard line-search Newton method, the number of times to access all data is

$$1 + \#\text{CG steps} + \#\text{line-search steps}. \tag{10}$$

The first term in (10) indicates that we need at least one full data access for calculating $f(\boldsymbol{w}^k)$ and $\nabla f(\boldsymbol{w}^k)$. In general, the second term is larger than the third, so the subsampled method aims at reducing the number of data accesses in the CG procedure by considering only a subset of data. Given that the whole data set must be accessed at least once, our ideas are to use one extra access of data for either improving the direction or reducing the number of line-search steps. Specifically, we take this extra data access to calculate and use the full Hessian matrix.

To begin, we consider the initial step size for line search. While in Byrd et al. [2011], they start with $\alpha_k = 1$, another setting is to consider

$$f(\boldsymbol{w}_k + \alpha \boldsymbol{d}_k) - f(\boldsymbol{w}_k) \approx \alpha \nabla f(\boldsymbol{w}_k)^T \boldsymbol{d}_k + \frac{1}{2}\alpha^2 \boldsymbol{d}_k^T H_k \boldsymbol{d}_k. \tag{11}$$

and minimize the right-hand side of (11) to get

$$\alpha = \frac{-\nabla f(\boldsymbol{w}_k)^T \boldsymbol{d}_k}{\boldsymbol{d}_k^T H_k \boldsymbol{d}_k}. \tag{12}$$

Using this value as the initial step size can potentially reduce the number of line search steps, but the extra cost is to calculate the product between the full Hessian and $\boldsymbol{d}_k$ in (12).

By still paying one extra data access, we extend the above idea to minimize (9) as much as possible. We propose minimizing the following form.

$$\min_{\beta_1,\beta_2} \ \frac{1}{2}(\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k)^T H_k (\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k) + \nabla f(\boldsymbol{w}_k)^T (\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k), \tag{13}$$

where $\boldsymbol{d}_k$ is the search direction obtained at the current iteration, and $\bar{\boldsymbol{d}}_k$ is a chosen vector. If $\bar{\boldsymbol{d}}_k = \mathbf{0}$, then problem (13) is reduced to (11) and the method in (12) becomes a special case here.

The function in (13) is convex to β_1 and β_2, so we can solve it by taking the gradient to be zero.

$$\begin{pmatrix} \boldsymbol{d}_k^T H_k \boldsymbol{d}_k & \bar{\boldsymbol{d}}_k^T H_k \boldsymbol{d}_k \\ \bar{\boldsymbol{d}}_k^T H_k \boldsymbol{d}_k & \bar{\boldsymbol{d}}_k^T H_k \bar{\boldsymbol{d}}_k \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} -\nabla f(\boldsymbol{w}_k)^T \boldsymbol{d}_k \\ -\nabla f(\boldsymbol{w}_k)^T \bar{\boldsymbol{d}}_k \end{pmatrix}. \tag{14}$$

This two-variable linear system has a simple closed-form solution. The resulting direction is a descent one because from (14),

$$\nabla f(\boldsymbol{w}_k)^T (\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k) = -(\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k)^T H_k (\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k) < 0. \tag{15}$$

Therefore, the backtracking line search is guaranteed to terminate.

Using this new direction, $\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k$, we have

$$\begin{aligned} & \nabla f(\boldsymbol{w}_k)^T(\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k) + \frac{1}{2}(\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k)^T H_k(\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k) \\ = & -\frac{1}{2}\nabla f(\boldsymbol{w}_k)^T(\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k) \\ \leq & -\eta \nabla f(\boldsymbol{w}_k)^T(\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k) \end{aligned}$$

if $\eta \leq 1/2$. Therefore, this modified subsampled Newton direction can easily satisfy the sufficient decrease condition (7) if (13) gives a good estimate of the function-value reduction.

Once (14) is solved, we must choose an initial step size for line search. Obviously we can apply (12) that aims to find a suitable initial α. Interestingly, the equality in (15) implies that if we apply $\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k$ to (12), then $\alpha = 1$ is obtained. This derivation indicates that a reasonable initial step size for line search is $\alpha = 1$.

For the first outer iteration, it is unclear what $\bar{\boldsymbol{d}}_1$ should be. We can simply set $\bar{\boldsymbol{d}}_1 = \mathbf{0}$, so β_1 is the same as that by (12).

In (14), two products between the full Hessian and vectors $\boldsymbol{d}_k, \bar{\boldsymbol{d}}_k$ are needed. However, with careful implementations, training instances are accessed once rather than twice; see an example in Section 5.1.

The remaining issue is the selection of the vector $\bar{\boldsymbol{d}}_k$. One possible candidate is $\boldsymbol{d}_{k-1}$, the direction at the previous iteration. Then information from both subsets S_{k-1} and S_k are used in generating the direction of the current iteration. Another possible $\bar{\boldsymbol{d}}_k$ is to use $-\nabla f(\boldsymbol{w}_k)$. Then (13) attempts to combine the second-order information (i.e., Newton direction $\boldsymbol{d}_k$) and the first-order information (i.e., negative gradient). In Section

Algorithm 2 An Improved Subsampled Hessian Newton-CG method

Given initial $\boldsymbol{w}_0 = \mathbf{0}$, $\text{CG}_{\max}$ and constants $\eta, \sigma \in (0, 1)$.

for $k = 0, 1, \ldots$ **do**

  Choose a subset S_k.

  Approximately solve (5) by CG to obtain a direction $\boldsymbol{d}_k$ after the condition (6) is satisfied or the number of CG iterations reaches $\text{CG}_{\max}$.

  if $k = 0$ **then**

    $\bar{\boldsymbol{d}}_k = \mathbf{0}$

  else

    Choose a suitable $\bar{\boldsymbol{d}}_k$

  end if

  Solve (13) to obtain a direction $\beta_{k,1}\boldsymbol{d}_k + \beta_{k,2}\bar{\boldsymbol{d}}_k$.

  Find α_k satisfying (7) by backtracking line search with initial $\alpha = 1$.

  Update $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \alpha_k(\beta_{k,1}\boldsymbol{d}_k + \beta_{k,2}\bar{\boldsymbol{d}}_k)$.

end for

6, we will compare different choices of $\bar{\boldsymbol{d}}_k$. A summary of our improved subsampled-Hessian Newton method is in Algorithm 2.

For problems with many variables (e.g., deep learning problems discussed in Section 5.4), it is difficult to solve (13) by using the full Hessian. Instead, we can replace H_k with H_{S_k} to have the following optimization problem.

$$\min_{\beta_1,\beta_2} \frac{1}{2}(\beta_1\boldsymbol{d}_k + \beta_2\bar{\boldsymbol{d}}_k)^T H_{S_k}(\beta_1\boldsymbol{d}_k + \beta_2\bar{\boldsymbol{d}}_k) + \nabla f(\boldsymbol{w}_k)^T(\beta_1\boldsymbol{d}_k + \beta_2\bar{\boldsymbol{d}}_k). \tag{16}$$

More details are shown in Section 5.4.

3.1 Relations with Prior Works of Using Second Directions

The concept of combining several search directions in one iteration has been applied in past optimization works. For example, in an early study by Polyak [1964], a heavy ball method was proposed so that

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \frac{\alpha}{L}\nabla f(\boldsymbol{w}_k) + \beta(\boldsymbol{w}_k - \boldsymbol{w}_{k-1}),$$

where α, β, and L are constants. It can be easily shown that

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \frac{1}{L}\sum_{i=0}^{k} \alpha\beta^{k-i}\nabla f(\boldsymbol{w}_k).$$

Thus, the update from $\boldsymbol{w}_k$ to $\boldsymbol{w}_{k+1}$ involves all past gradients $\nabla f(\boldsymbol{w}_0), \ldots, \nabla f(\boldsymbol{w}_k)$. In Drori and Teboulle [2014], they have shown that some other first-order optimization methods can be expressed in a similar setting of using all past gradients.

The main difference between ours and past studies is that our directions ($\boldsymbol{d}_k$ and if $\bar{\boldsymbol{d}}_k = \boldsymbol{d}_{k-1}$) are obtained from using second-order information. The coefficients for combining directions are then obtained by solving a two-variable optimization problem.

4 Convergence

In this section, we discuss the convergence properties of the proposed methods. The proof is related to that in Byrd et al. [2011], but some essential modifications are needed. In addition, our analysis more broadly covers continuously differentiable $f(\boldsymbol{w})$ because Byrd et al. [2011] require twice differentiability. We begin with proving the convergence of using (13) because the proof for (16) is similar.

4.1 Convergence of Using (13)

Following Byrd et al. [2011], we need H_{S_k} to be uniformly positive definite. That is, there exists $m > 0$ such that

$$\boldsymbol{v}^T H_{S_k} \boldsymbol{v} \geq m||\boldsymbol{v}||^2, \ \forall \boldsymbol{v} \in R^n. \tag{17}$$

If a convex loss is used, this property can be easily achieved by adding an l2 regularization term to the objective function. We also need that there exists $M > 0$ such that

$$||H_{S_k}|| \leq M, \ \forall k. \tag{18}$$

To connect H_{S_k} and H_k, as mentioned in Section 1, we require that there exist $\bar{m}$ and $\bar{M}$ such that

$$\bar{m} H_{S_k} \preceq H_k \preceq \bar{M} H_{S_k}, \forall k. \tag{19}$$

We present the convergence result in the following theorem.

Theorem 1. *Let $f(\boldsymbol{w})$ be continuously differentiable and assume the following conditions hold.*

1. The sublevel set $\{\boldsymbol{w} : \ f(\boldsymbol{w}) \leq f(\boldsymbol{w}_0)\}$ is bounded, where $\boldsymbol{w}_0$ is the initial point.

2. $\nabla f(\boldsymbol{w})$ is Lipschitz continuous. That is, there exists $L > 0$ such that

$$||\nabla f(\boldsymbol{w}) - \nabla f(\hat{\boldsymbol{w}})|| \leq L||\boldsymbol{w} - \hat{\boldsymbol{w}}||, \ \forall \boldsymbol{w}, \ \hat{\boldsymbol{w}}.$$

3. (17)*,* (18) *and* (19) *hold.*

Then, the sequence $\{\boldsymbol{w}_k\}$ generated by Algorithm 2 using the direction of solving (13) *satisfies*

$$\lim_{k \to \infty} \nabla f(\boldsymbol{w}_k) = \boldsymbol{0}.$$

Proof. Like Byrd et al. [2011], we establish all needed conditions in Theorem 11.7 of Griva et al. [2009] for the convergence proof. We begin with showing that

$$-\frac{(\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k)^T \nabla f(\boldsymbol{w}_k)}{||\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k|| ||\nabla f(\boldsymbol{w}_k)||} \tag{20}$$

is bounded above zero for all k. From (15), (19) and (17),

$$\begin{aligned} -\nabla f(\boldsymbol{w}_k)^T(\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k) &= (\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k)^T H_k (\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k) \\ &\geq \bar{m}(\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k)^T H_{S_k} (\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k) \\ &\geq \bar{m} m ||\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k||^2. \end{aligned} \tag{21}$$

Following Byrd et al. [2011] to apply properties of CG, there exists a matrix Q_k and a vector $\boldsymbol{v}_k$ such that

$$\boldsymbol{d}_k = Q_k \boldsymbol{v}_k \quad \text{and} \quad Q_k^T H_{S_k} Q_k \boldsymbol{v}_k = -Q_k^T \nabla f(\boldsymbol{w}_k), \tag{22}$$

where Q_k's columns form a orthonormal basis of the Krylov subspace that includes $\nabla f(\boldsymbol{w}_k)$.[2] Therefore, $\nabla f(\boldsymbol{w}_k)$ is in the range of Q_k's columns, and hence

$$||Q_k^T \nabla f(\boldsymbol{w}_k)|| = ||\nabla f(\boldsymbol{w}_k)||. \tag{23}$$

From (23), (22) and (18), we can derive

$$||\nabla f(\boldsymbol{w}_k)|| = ||Q_k^T \nabla f(\boldsymbol{w}_k)|| \leq ||H_{S_k}|| ||Q_k \boldsymbol{v}_k|| \leq M ||\boldsymbol{d}_k||. \tag{24}$$

[2] This property holds because the initial point of the CG procedure is the zero vector.

Using (18) and (19),

$$\begin{aligned}
M||\boldsymbol{d}_k||||\beta_1\boldsymbol{d}_k+\beta_2\bar{\boldsymbol{d}}_k|| &\geq ||\boldsymbol{d}_k||||H_{S_k}||||\beta_1\boldsymbol{d}_k+\beta_2\bar{\boldsymbol{d}}_k|| \\
&\geq \frac{1}{\bar{M}}||\boldsymbol{d}_k||||H_k||||\beta_1\boldsymbol{d}_k+\beta_2\bar{\boldsymbol{d}}_k|| \\
&\geq \frac{1}{\bar{M}}\boldsymbol{d}_k^T H_k(\beta_1\boldsymbol{d}_k+\beta_2\bar{\boldsymbol{d}}_k) \\
&= -\frac{1}{\bar{M}}\nabla f(\boldsymbol{w}_k)^T\boldsymbol{d}_k \qquad (25) \\
&= \frac{1}{\bar{M}}\boldsymbol{v}_k^T Q_k^T H_{S_k} Q_k\boldsymbol{v}_k \qquad (26) \\
&= \frac{1}{\bar{M}}\boldsymbol{d}_k^T H_{S_k}\boldsymbol{d}_k \geq \frac{m}{\bar{M}}||\boldsymbol{d}_k||^2, \qquad (27)
\end{aligned}$$

where (25) is from the first equality in (14), (26) is from (22), and (27) is from (17). Therefore, (21), (24) and (27) imply that

$$-\frac{(\beta_1\boldsymbol{d}_k+\beta_2\bar{\boldsymbol{d}}_k)^T\nabla f(\boldsymbol{w}_k)}{||\nabla f(\boldsymbol{w}_k)||||\beta_1\boldsymbol{d}_k+\beta_2\bar{\boldsymbol{d}}_k||} \geq \frac{\bar{m}m||\beta_1\boldsymbol{d}_k+\beta_2\bar{\boldsymbol{d}}_k||}{M||\boldsymbol{d}_k||} \geq \frac{m^2}{M^2}\frac{\bar{m}}{\bar{M}},$$

which means our modified direction $\beta_1\boldsymbol{d}_k+\beta_2\bar{\boldsymbol{d}}_k$ is a sufficient descent direction.

The remaining condition needed is to show that the search direction is gradient related and is bounded in norm. From (27) and (24), we have

$$||\beta_1\boldsymbol{d}_k+\beta_2\bar{\boldsymbol{d}}_k|| \geq \frac{m}{M\bar{M}}||\boldsymbol{d}_k|| \geq \frac{m}{M^2\bar{M}}||\nabla f(\boldsymbol{w}_k)||.$$

From (21),

$$||\beta_1\boldsymbol{d}_k+\beta_2\bar{\boldsymbol{d}}_k||^2 \leq \frac{-\nabla f(\boldsymbol{w}_k)^T(\beta_1\boldsymbol{d}_k+\beta_2\bar{\boldsymbol{d}}_k)}{m\bar{m}} \leq \frac{||\nabla f(\boldsymbol{w}_k)||||\beta_1\boldsymbol{d}_k+\beta_2\bar{\boldsymbol{d}}_k||}{m\bar{m}},$$

so

$$||\beta_1\boldsymbol{d}_k+\beta_2\bar{\boldsymbol{d}}_k|| \leq \frac{||\nabla f(\boldsymbol{w}_k)||}{m\bar{m}}. \qquad (28)$$

Because the sublevel set is bounded, from the continuity of $||\nabla f(\boldsymbol{w}_k)||$, we have that $||\nabla f(\boldsymbol{w}_k)||$ is bounded, and so is $||\beta_1\boldsymbol{d}_k+\beta_2\bar{\boldsymbol{d}}_k||$.

Finally, we have shown that all conditions in Theorem 11.7 of Griva et al. [2009] are satisfied, so the convergence is obtained. □

It can be seen that (21), (24) and (27) are three major inequalities derived in the proof. While (24) is the same as that in Byrd et al. [2011], (21) and (27) are new. The key difference is that we must make a connection between $||\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k||$ and $||\boldsymbol{d}_k||$.

4.2 Convergence of Using (16)

We show that the convergence proof in Section 4.1 can be modified if (16) is used. Because (16) differs from (13) only in using H_{S_k} rather than H_k, all we must address are places in Theorem 1 that involve H_k. Clearly we only need to check inequalities (21) and (27). Easily we see that they still hold and the derivation is in fact simpler. Therefore, the convergence is established.

5 Examples: Logistic Regression, l2-loss Linear SVM, Maximum Entropy and Deep Neural Networks

In this section, we discuss how the proposed approach can be applied to various machine learning problems.

5.1 Logistic Regression

For two-class data with label $y \in \{+1, -1\}$, the logistic regression (LR) loss is as follows:

$$\xi(\boldsymbol{w}; \boldsymbol{x}, y) = \log(1 + e^{-y\boldsymbol{w}^T\boldsymbol{x}}).$$

Furthermore, the optimization problem of regularized logistic regression is

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}), \text{ where } f(\boldsymbol{w}) \equiv \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{C}{l}\sum_{i=1}^{l}\log(1+e^{-y_i\boldsymbol{w}^T\boldsymbol{x}_i}).$$

Note that $f(\boldsymbol{w})$ is twice continuously differentiable with the following Hessian.

$$\nabla^2 f(\boldsymbol{w}) = I + \frac{C}{l}X^T D X, \tag{29}$$

where I is the identity matrix, $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l]^T$, and D is a diagonal matrix with

$$D_{ii} = \frac{e^{-y_i\boldsymbol{w}^T\boldsymbol{x}_i}}{(1+e^{-y_i\boldsymbol{w}^T\boldsymbol{x}_i})^2},\ i = 1, \ldots, l.$$

By using a subset S_k, the subsampled Hessian-vector product is

$$H_{S_k}\boldsymbol{v} = \boldsymbol{v} + \frac{C}{|S_k|}(X^T(D_{S_k}(X\boldsymbol{v}))). \tag{30}$$

By a sequence of matrix-vector products in (30), we have a Hessian-free approach because H_{S_k} is never explicitly formed.

After the CG procedure, we must calculate $H_k\boldsymbol{d}_k$ and $H_k\bar{\boldsymbol{d}}_k$ in order to apply our proposed approach. This step may be the bottleneck because the whole training set rather than a subset is used. From (30), we can calculate $X\boldsymbol{d}_k$ and $X\bar{\boldsymbol{d}}_k$ together, so the number of data accesses remains the same as using only $\boldsymbol{d}_k$.[3]

For convergence, we check if assumptions in Theorem 1 hold.

(1) Because of the regularization term $\boldsymbol{w}^T\boldsymbol{w}/2$, the sublevel set is bounded. See, for example, Appendix A of Lin et al. [2008].

[3] Note that from the recent development of linear classification, it is known that the number of data accesses may affect the running time more than the number of operations.

(2) The gradient is Lipschitz continuous. The mean-value theorem implies

$$||\nabla f(\boldsymbol{w}) - \nabla f(\hat{\boldsymbol{w}})|| \le ||\nabla^2 f(\bar{\boldsymbol{w}})||||\boldsymbol{w} - \hat{\boldsymbol{w}}||,$$

where $\bar{\boldsymbol{w}}$ is between $\boldsymbol{w}$ and $\hat{\boldsymbol{w}}$. From (29) and $D_{ii} \le 1$, $||\nabla^2 f(\bar{\boldsymbol{w}})||$ is bounded by

$$\begin{aligned} ||\nabla^2 f(\boldsymbol{w})|| &\le 1 + \frac{C}{l}||X^T||||D||||X|| \\ &\le 1 + \frac{C}{l}||X^T||||X||. \end{aligned}$$

(3) The regularization term implies that (17) holds with $m = 1$. For (18), because $D_{ii} \le 1$, we have

$$1 \le ||H_{S_k}|| \le 1 + \frac{C}{l}||X^T||||D||||X|| \le 1 + \frac{C}{l}||X^T||||X||. \tag{31}$$

Next, we check (19). Because H_k is a special case of H_{S_k} when $S_k = \{1, \ldots, l\}$, it also satisfies (17) and (18). Then, (19) follows.

5.2 l2-loss Linear SVM

For two-class data with label $y \in \{+1, -1\}$, the squared hinge loss (l2-loss) takes the following form.

$$\xi(\boldsymbol{w}; \boldsymbol{x}, y) = \max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x})^2.$$

Two-class l2-loss SVM then solves the following optimization problem.

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}), \text{ where } f(\boldsymbol{w}) \equiv \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{C}{l}\sum_{i \in B}(1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i)^2. \tag{32}$$

In (32), $y_i \in \pm 1, i = 1, \ldots, l$, C is the regularization parameter, and $B = \{i \mid 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i > 0\}$. It is known that (32) is differentiable but not twice differentiable. However, we can define the following generalized Hessian [Mangasarian, 2006],

$$\nabla^2 f(\boldsymbol{w}) = I + 2\frac{C}{l}X^T D X, \tag{33}$$

where I is the identity matrix, $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_l]^T$, and D is a diagonal matrix with

$$D_{ii} = \begin{cases} 1 & \text{if } i \in B, \\ 0 & \text{otherwise.} \end{cases}$$

Then the Newton method can still be applied; see an example in Lin et al. [2008] that uses the full Hessian. For the subsampled Hessian, the diagonal matrix D is modified to have

$$D_{ii} = \begin{cases} 1 & \text{if } i \in B \cap S_k, \\ 0 & \text{otherwise.} \end{cases}$$

Regarding the convergence, most assumptions in Theorem 1 hold by the same explanation in Section 5.1. The only exception is the Lipschitz continuity of $\nabla f(\boldsymbol{w})$, where a proof can be found in Eq. (15) of Mangasarian [2002].

5.3 Maximum Entropy

For multi-class data with label $y \in \{1, \ldots, k\}$, maximum entropy (ME) is an extension of LR.[4] The probability that an instance $\boldsymbol{x}$ is associated with label y is

$$\mathcal{P}_{\boldsymbol{w}}(y|\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_y^T \boldsymbol{x})}{\sum_{c=1}^{k} \exp(\boldsymbol{w}_c^T \boldsymbol{x})},$$

where y is the label of $\boldsymbol{x}$ and $\boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_k \end{bmatrix} \in R^{kn \times 1}$.

Minimizing the negative log likelihood with a regularization term leads to the following optimization problem.

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}), \quad \text{where } f(\boldsymbol{w}) \equiv \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{C}{l}\sum_{i=1}^{l}(\log(\sum_{c=1}^{k}\exp(\boldsymbol{w}_c^T\boldsymbol{x}_i)) - \boldsymbol{w}_{y_i}^T\boldsymbol{x}_i).$$

[4] For a detailed explanation connecting LR and ME, see, for example, Section 5.2 of Huang et al. [2010].

In Byrd et al. [2011], ME is referred to as multinomial logistic regression.

Notice that $f(\boldsymbol{w})$ is twice continuously differentiable. The Hessian-vector product of $f(\boldsymbol{w})$ is

$$H\boldsymbol{v} = \begin{bmatrix} (H\boldsymbol{v})^1 \\ \vdots \\ (H\boldsymbol{v})^k \end{bmatrix}, \text{ where } \boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_1 \\ \vdots \\ \boldsymbol{v}_k \end{bmatrix} \tag{34}$$

and

$$P_{i,s} = \frac{\exp(\boldsymbol{w}_s^T \boldsymbol{x}_i)}{\sum_{c=1}^k \exp(\boldsymbol{w}_c^T \boldsymbol{x}_i)},$$
$$(H\boldsymbol{v})^t = \boldsymbol{v}_t + \frac{C}{l} \sum_{i=1}^{l} \big[P_{i,t}(\boldsymbol{v}_t^T \boldsymbol{x}_i - \sum_{c=1}^{k} P_{i,c} \boldsymbol{v}_c^T \boldsymbol{x}_i) \big] \boldsymbol{x}_i.$$

Details of the derivation are in the supplementary materials. If using the subsampled Hessian, we have

$$(H_{S_k}\boldsymbol{v})^t = \boldsymbol{v}_t + \frac{C}{|S_k|} \sum_{i \in S_k} \big[P_{i,t}(\boldsymbol{v}_t^T \boldsymbol{x}_i - \sum_{c=1}^{k} P_{i,c} \boldsymbol{v}_c^T \boldsymbol{x}_i) \big] \boldsymbol{x}_i.$$

Regarding the convergence, we prove that Theorem 1 holds, but leave details in the supplementary materials.

5.4 Deep Neural Networks

We apply deep neural networks for multi-class classification, where the number of classes is k and the class labels are assumed to be $1, \ldots, k$. A deep neural network maps each feature vector to one of the class labels by the connection of nodes in a multi-layer structure. Between two layers a weight vector maps inputs (the previous layer) to outputs (the next layer). An illustration is in Figure 2.

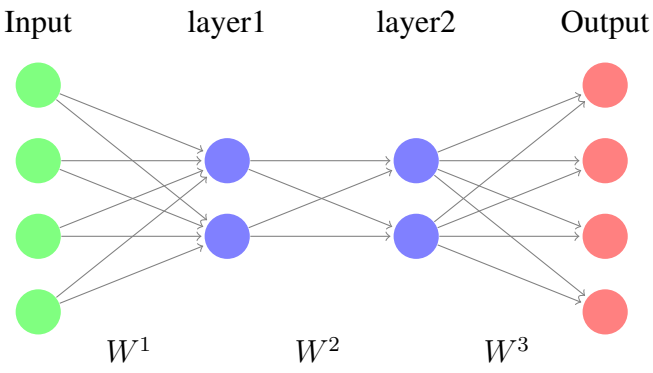


Figure 2: An illustration of the deep neural networks, where W^1 contains 8 elements, W^2 contains 4, and W^3 has 8.

Assume the network has L layers. We use n_0-$\cdots$-n_L to denote a structure so that the number of neurons at the mth layer is n_m.[5] The example in Figure 2 has a structure of 4-2-2-4. At the mth layer, the output $\boldsymbol{z}^m$ is obtained by the following recurrence.

$$\begin{aligned} \boldsymbol{x}^m &= W^m \boldsymbol{z}^{m-1}, \\ \boldsymbol{z}_i^m &= \sigma(\boldsymbol{x}_i^m), \ \forall i, \end{aligned} \tag{35}$$

where w_{ij}^m is the weight from node i of the $(m-1)$st layer to node j of the mth layer, σ is an activation function,[6] $\boldsymbol{z}^L = \boldsymbol{x}^L$ is the output, and $\boldsymbol{z}^0 = \boldsymbol{x}^0 = \boldsymbol{x}$ is the given input feature vector. Assume $\boldsymbol{\theta} = [W^1; W^2; \ldots; W^L]$ is the collection of all variables. A model is obtained by minimizing the following regularized training losses:

$$\min_{\boldsymbol{\theta}} \ f(\boldsymbol{\theta}), \ \text{where } f(\boldsymbol{\theta}) = \frac{1}{2C}||\boldsymbol{\theta}||^2 + \frac{1}{l}\sum_{i=1}^{l} \xi(\boldsymbol{\theta}; \boldsymbol{x}_i, y_i), \tag{36}$$

where $C > 0$ is a regularization constant. We follow Martens [2010] to consider the

[5] Note that n_0 is the number of features and $n_L = k$ is the number of classes.

[6] In this paper, we use the sigmoid function as an activation function. That is, $1/(1+e^{-x})$.

least-square loss:

$$\xi(\boldsymbol{\theta}; \boldsymbol{x}, y) = ||\boldsymbol{e}_y - \boldsymbol{z}^L||^2,$$

where $\boldsymbol{e}_y = [\underbrace{0, \ldots, 0}_{1 \text{ to } y-1}, 1, 0, \ldots, 0]^T$ and $y \in \{1, \ldots, k\}$. After the training procedure of constructing the network, the predicted class for a test instance $\boldsymbol{x}$ is

$$\arg \max_{j \in \{1,\ldots,n_L\}} \quad \boldsymbol{z}_j^L.$$

Because the total number of parameters $\sum_{t=1}^{L} n_{t-1} n_t$ is large, to apply Newton methods, we need a Hessian-free approach. Two major challenges are

1. In contrast to (30), the (sub)-Hessian vector product is now much more complicated because of the network structure.

2. $f(\boldsymbol{\theta})$ is not a convex function.

Martens [2010] and Martens and Sutskever [2012] have designed a subsampled Hessian Newton method to handle these two difficulties. For fast Hessian-vector product, they employ the technique of "forward-differentiation" [Wengert, 1964, Pearlmutter, 1994]. We give more details in the supplementary materials.

For a non-convex $f(\boldsymbol{\theta})$, H_k is not positive semi-definite, so Schraudolph [2002] has proposed using a generalized Gauss-Newton matrix to approximate the Hessian in deep learning problems. Though we do not give details, a generalized Gauss-Newton matrix is positive semi-definite and takes the following form

$$G \equiv J^T B J, \tag{37}$$

where B is a positive definite matrix. We can clearly see that this form is similar to the Hessian matrix of LR and l2-loss SVM in (29) and (33), respectively. We can

further add a regularization term to ensure the positive definiteness. The Hessian-vector product becomes the product between the generalized Gauss-Newton matrix and the vector. This calculation can be conducted by $\mathcal{R}$-operators discussed above.

In Algorithm 1, the line search procedure is used to ensure the convergence. Other optimization techniques for a similar purpose include Levenberg-Marquardt method [Moré, 1978] and trust region methods [Conn et al., 2000]. Martens [2010] applies the Levenberg-Marquardt method prior to the line search procedure. Specifically, instead of the linear system in (5), the direction $\boldsymbol{d}_k$ is obtained by solving the following linear system:

$$(G_{S_k} + \lambda_k I)\boldsymbol{d}_k = -\nabla f(\boldsymbol{\theta}_k),$$

where G_k is the Gauss-Newton matrix in (37) at the current iteration, I is an identity matrix, and λ_k is a damping factor. For the next iteration, λ_{k+1} is updated by

$$\lambda_{k+1} = \begin{cases} \lambda_k \times \text{drop} & \text{if } \rho_k > 0.75, \\ \lambda_k & \text{if } 0.25 < \rho_k < 0.75, \\ \lambda_k \times \text{boost} & \text{otherwise.} \end{cases}$$

Note that $(\text{drop}, \text{boost})$ are constants and

$$\rho_k = \frac{f(\boldsymbol{\theta}_k + \boldsymbol{d}_k) - f(\boldsymbol{\theta}_k)}{q_k(\boldsymbol{d}_k)},$$

where

$$q_k(\boldsymbol{d}) \equiv \nabla f(\boldsymbol{\theta}_k)^T \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T G_{S_k} \boldsymbol{d} \tag{38}$$

and ρ_k is the ratio between actual and predicted function reductions. According to Martens and Sutskever [2012], this adjustment of the direction is practically useful in training deep networks.

Next we discuss how to improve the direction using methods developed in Section 3. Because deep neural networks have a large number of variables, we solve (16) instead of (13). Further, H_{S_k} in (16) is replaced with the Gauss-Newton matrix G_k. The optimization problem is convex in β_1 and β_2, so the following linear system is solved.

$$\begin{pmatrix} \boldsymbol{d}_k^T G_{S_k} \boldsymbol{d}_k & \bar{\boldsymbol{d}}_k^T G_{S_k} \boldsymbol{d}_k \\ \bar{\boldsymbol{d}}_k^T G_{S_k} \boldsymbol{d}_k & \bar{\boldsymbol{d}}_k^T G_{S_k} \bar{\boldsymbol{d}}_k \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} -\nabla f(\boldsymbol{\theta}_k)^T \boldsymbol{d}_k \\ -\nabla f(\boldsymbol{\theta}_k)^T \bar{\boldsymbol{d}}_k \end{pmatrix}. \tag{39}$$

One may criticize that $\boldsymbol{d}_k$ is already the best solution of minimization the second-order approximation $q_k(\boldsymbol{d})$ defined in (38), so (16) should not give a better direction. However, because of the damping factor λ_k, $\boldsymbol{d}_k$ is not the optimal solution to minimize $q_k(\boldsymbol{d})$ in (38). Therefore, (39) may be useful to find a more accurate solution to minimize $q_k(\boldsymbol{d})$ and hence obtain a better direction. We will see detailed experiments in Section 6.2.

Because the objective function of deep networks is non-convex and our algorithm has been extended to incorporate the LM procedure, we do not have theoretical convergence like that in Section 4.

6 Experiments

In this section, we conduct experiments on logistic regression, l2-loss linear SVM, maximum entropy, and deep neural networks. We compare the proposed approaches with existing sub-sampled Hessian Newton methods in Byrd et al. [2011] and Martens [2010].

Programs for experiments in this paper can be found at `http://www.csie.ntu.edu.tw/~cjlin/papers/sub_hessian/sub_hessian_exps.tar.`

Table 1: Summary of two-class sets used for experiments on logistic regression and l2-loss linear SVM. n is the number of features, and $\bar{n}$ is the average number of non-zero features per training example. l is the number of training examples. C^*_{lr} and C^*_{l2} are the parameters among $\{2^{-6}, 2^{-5}, \ldots, 2^6\}$ to achieve the best five-fold cross validation accuracy for logistic regression and l2-loss linear SVM, respectively.

Data set	n	$\bar{n}$	l	C^*_{lr}	C^*_{l2}
news20	1,355,191	455	19,996	64l	64l
yahoo-korea	3,052,939	340	368,444	32l	2l
kdd2010-a	20,216,830	36	8,407,752	0.0625l	0.015625l
kdd2010-b	29,890,095	29	19,264,097	0.0625l	0.015625l

gz. All data sets except yahoo-japan are publicly available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

6.1 Logistic Regression and l2-loss Linear SVM

We select some large, sparse, and two-class data for experiments. Such data sets are commonly used in evaluating linear classifiers such as SVM and logistic regression. Detailed data statistics are in Tables 1.

We compare the following five methods.

1. *Full*: the Newton method of using the full Hessian. We do not use $\text{CG}_{\max}$ (the maximal number of CG iterations) for a stopping condition of the CG procedure, so only (6) is used.

2. *Full-CG*: the Newton method of using the full Hessian. $\text{CG}_{\max}$ is set as the maximal

number of CG steps. For example, *Full-CG10* means that $\text{CG}_{\max} = 10$.

3. *Subsampled*: the method proposed in Byrd et al. [2011], where the backtracking line search starts with $\alpha = 1$. See also Algorithm 1.

4. *Method 1*: the same as *Subsampled*, but the initial α for backtracking line search is by (12). Although this modification is minor, we are interested in checking how important the initial α of line search is in a subsampled Newton method.

5. *Method 2*: the method proposed in Section 3 by using the direction $\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k$. Note that we set $\bar{\boldsymbol{d}}_k = \boldsymbol{d}_{k-1}$.

For the CG procedure at each iteration, we let $\sigma = 0.1$ in the stopping condition (6). Following Byrd et al. [2011], except *Full* we set

$$\text{CG}_{\max} = 10$$

because in some situations (6) is difficult to be satisfied.

The constant η in the sufficient decrease condition is chosen to be 0.0001. The ratio of backtracking line search is $1/2$; see (8). Our experimental framework is modified from the Newton-CG implementation of the software LIBLINEAR [Fan et al., 2008].

We present the running time in Figure 3 (logistic regression) and Figure 4 (l2-loss linear SVM) by using the following subsampling rates:

1. $\frac{|S_k|}{l} = 5\%$.

2. $\frac{|S_k|}{l} = 1\%$.

In each sub-figure, the x-axis shows the running time in seconds, while the y-axis gives the relative difference to the optimal value defined as

$$\frac{f(\boldsymbol{w}_k) - f(\boldsymbol{w}^*)}{f(\boldsymbol{w}^*)}, \tag{40}$$

where $\boldsymbol{w}^*$ is an optimal solution and $\boldsymbol{w}_k$ is the k-th iterate. Because $\boldsymbol{w}^*$ is not available, we run one method long enough to obtain an approximate reference value. Note that in Figures 3 and 4, both x-axis and y-axis are log-scaled.

We can make the following observations.

1. Among the three subsampled Hessian Newton methods, in general

   $$\textit{Method 2} > \textit{Method 1} > \textit{Subsampled},$$

   where "$>$" indicates faster convergence speed. The difference is bigger when using a smaller S_k. In this situation, the obtained direction using subsampled Hessian is very different from the full Newton direction, so some adjustments are very useful. Overall our approach (*Method 2*) significantly improves upon the approach in Byrd et al. [2011].

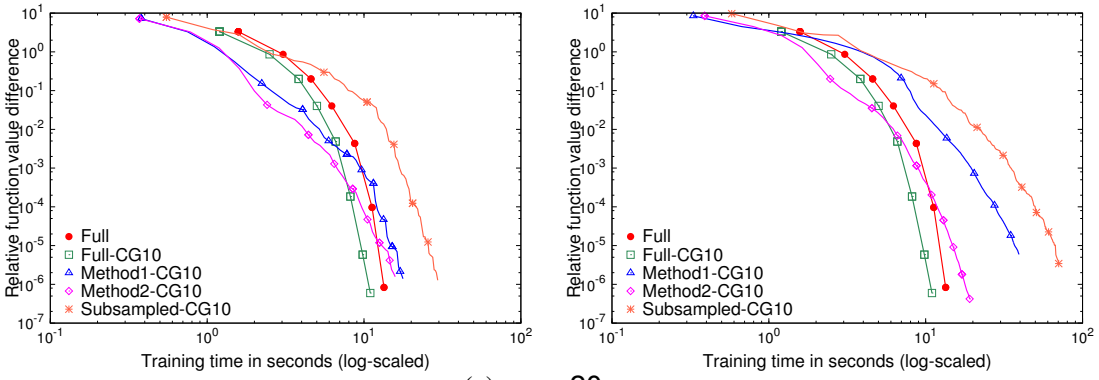
   In Figures 4(c) and 4(d), *Method 1* is slightly worse than *Subsampled*. An investigation shows that at final iterations, the sufficient decrease condition (7) holds at $\alpha_k = 1$. In such a situation, the effort by *Method 1* to find a good initial α is not very cost-effective.

2. When $|S_k|/l$ is reduced from 5% to 1%, the running time of the three subsampled Newton methods (*Subsampled*, *Method 1*, *Method 2*) increases. This result is expected because a smaller $|S_k|$ leads to worse directions.

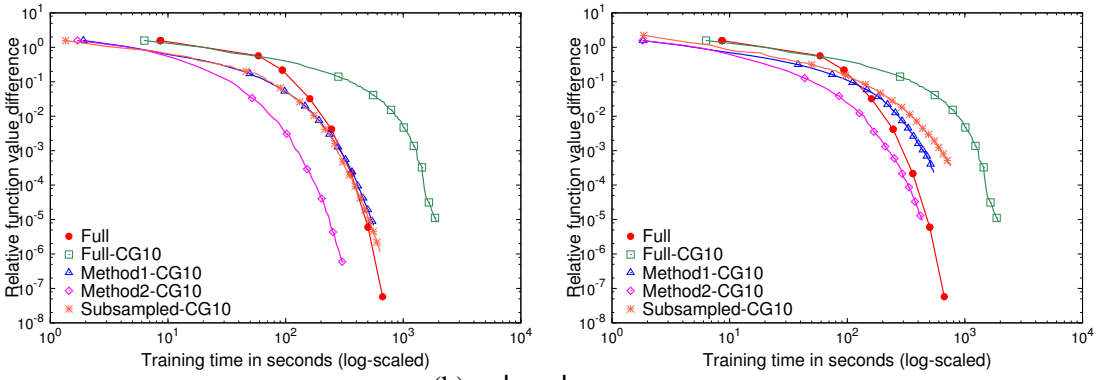
3. A comparison between *Full* and *Full-CG10* shows that the number of CG steps per outer iteration may significantly affect the running time. A smaller $\text{CG}_{\max}$ reduces the cost per outer iteration, but may cause more outer iterations. In Figures 3 and 4, *Full-CG10* is in general slower than *Full*, so selecting a larger $\text{CG}_{\max}$ seems to be necessary for these problems. On the other hand, except in Figures 3(a) and 4(a), *Subsampled-CG10* is faster than *Full-CG10*. This result is consistent with that in Byrd et al. [2011]. However, *Subsampled-CG10* is slower than *Full*, where they differ not only in the use of subsampled or full Hessian, but also in the stopping condition of the CG procedure. This example indicates that in comparing two methods, it is important to keep all settings except the one for analysis to be the same.

4. After using our proposed techniques, *Method 2* becomes faster than *Full* and *Full-CG10*. The only exception is news20, which has #features $\gg$ #instances, so using only a subset S_k may cause a significant information loss. Therefore, subsampled Newton methods are less suitable for such data sets. In addition, the C value chosen for this set is relatively large. The Hessian matrix becomes more ill-conditioned in this situation, so using full Hessian can obtain better search directions.

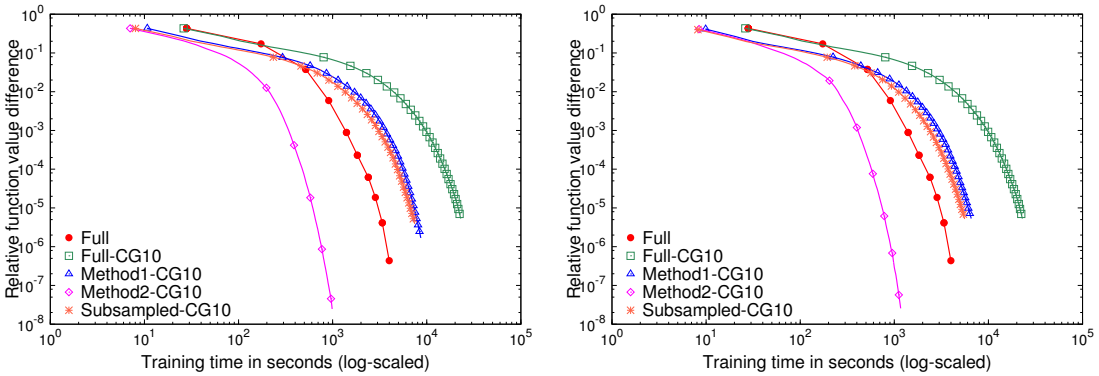
The above discussion indicates the importance of setting a proper $\text{CG}_{\max}$ value, so in Figure 5, we analyze results of using various $\text{CG}_{\max}$ values. In Figure 5(a), $\text{CG}_{\max} = 10$ is the best, but in Figure 5(b), the best $\text{CG}_{\max}$ becomes 100. Therefore, the best $\text{CG}_{\max}$ value is problem dependent, regardless of using full or subsampled Hessian. Unfortunately, we do not have a good strategy for selecting a suitable $\text{CG}_{\max}$, so this is a future issue for investigation.



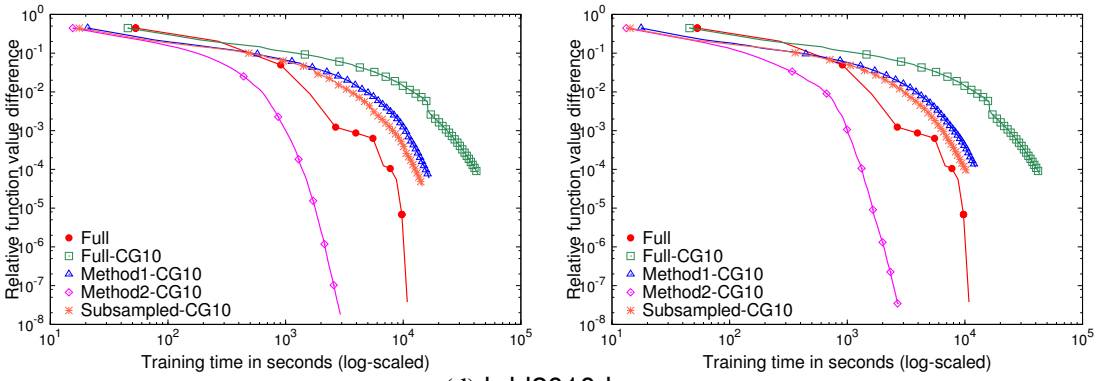
(a) news20



(b) yahoo-korea

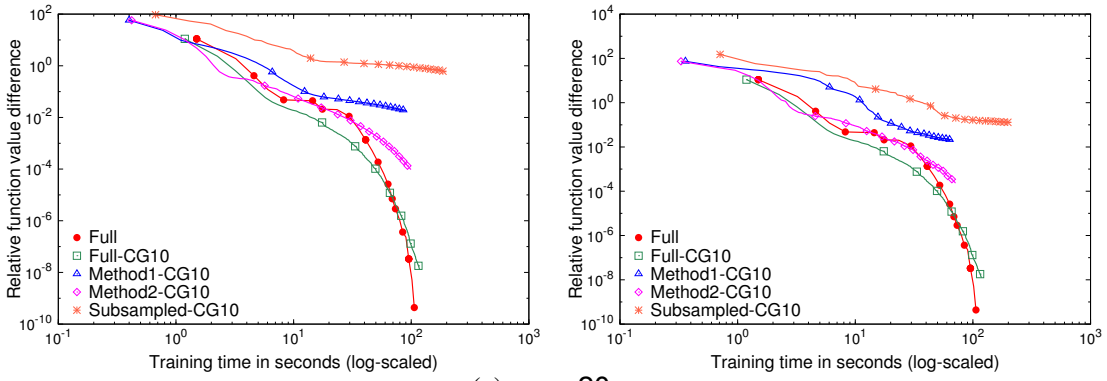


(c) kdd2010-a

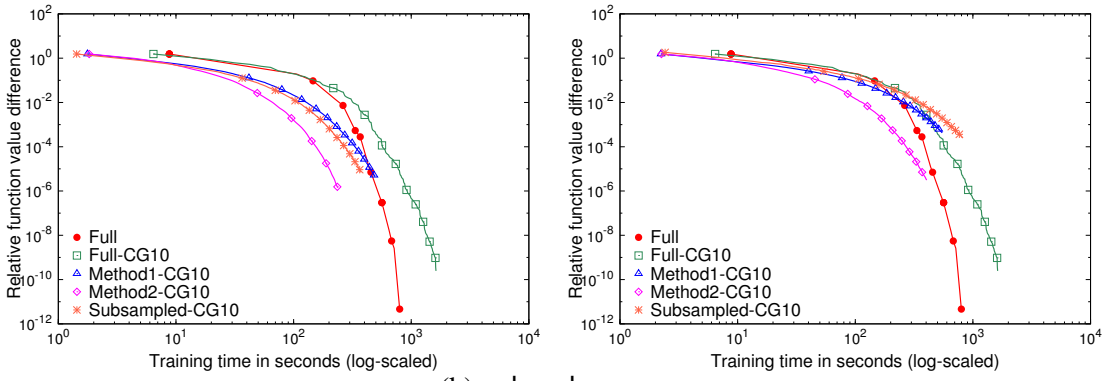


(d) kdd2010-b

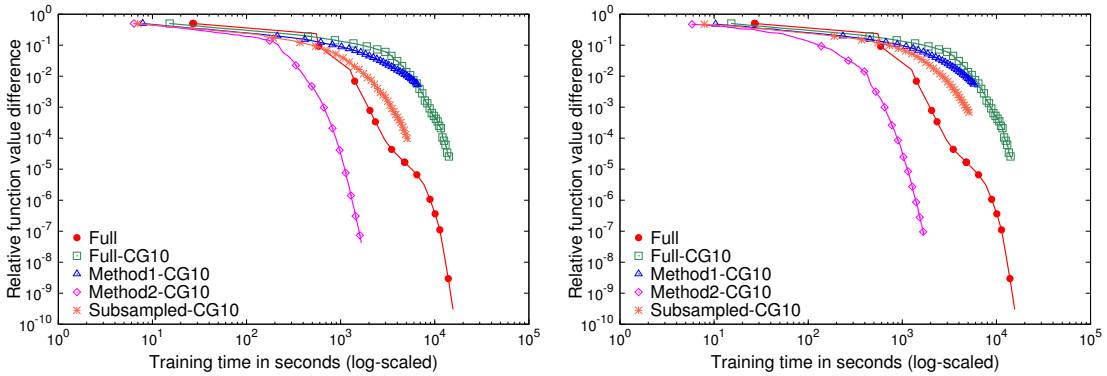
Figure 3: Experiments on logistic regression. We present running time (in seconds) versus the relative difference to the optimal function value. Both x-axis and y-axis are log-scaled. Left: $|S_k|/l = 5\%$. Right: $|S_k|/l = 1\%$.



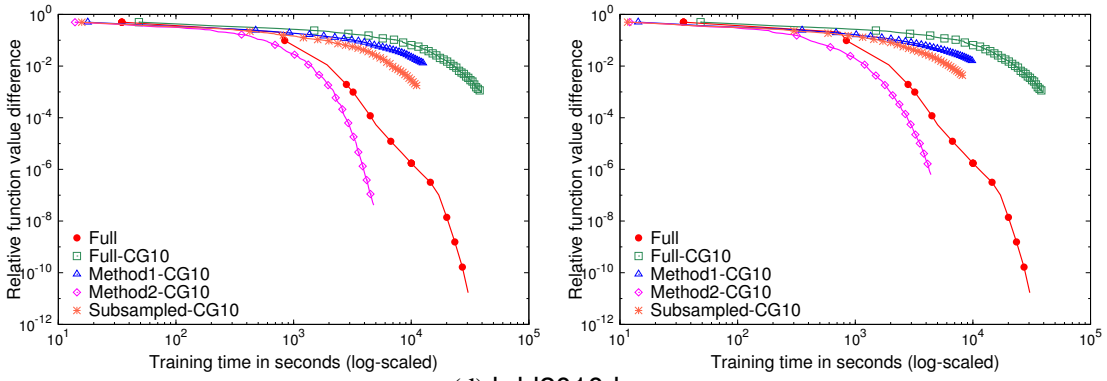
(a) news20



(b) yahoo-korea

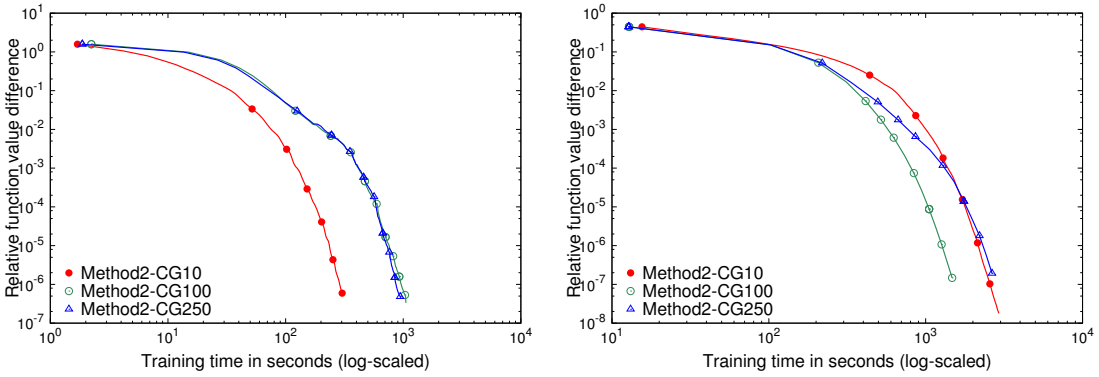


(c) kdd2010-a



(d) kdd2010-b

Figure 4: Experiments on L2-loss linear SVM. We present running time (in seconds) versus the relative difference to the optimal function value. Both x-axis and y-axis are log-scaled. Left: $|S_k|/l = 5\%$. Right: $|S_k|/l = 1\%$.



(a) yahoo-korea (b) kdd2010-b

Figure 5: A comparison between different $\text{CG}_{\max}$. Data sets yahoo-korea and kdd2010-b, 5% subsampled Hessian, and logistic regression (LR) are considered. We present running time (in seconds) versus the relative difference to the optimal function value. Both x-axis and y-axis are log-scaled.

An important issue of the proposed method in Section 3 is the selection of $\bar{\boldsymbol{d}}_k$. While we have experimentally demonstrated that $\bar{\boldsymbol{d}}_k = \boldsymbol{d}_{k-1}$ is useful, we have also checked

$$\bar{\boldsymbol{d}}_k = -\nabla f(\boldsymbol{w}_k).$$

Because of space consideration, we leave details in the supplementary materials. Results clearly show that using $\bar{\boldsymbol{d}}_k = \boldsymbol{d}_{k-1}$ is much better than $-\nabla f(\boldsymbol{w}_k)$. We believe the superiority of $\boldsymbol{d}_{k-1}$ is because it comes from solving a sub-problem of using some second-order information.

6.2 Maximum Entropy for Multi-Class Classification

We select some large multi-class data for experiments. Detailed data statistics are in Tables 2. For simplicity, $C = l$ is used for all problems.

Table 2: Summary of multi-class data sets used for experiments on maximum entropy. n is the number of features, and $\bar{n}$ is the average number of non-zero features per training example. l is the number of training examples. For aloi, we use only 80% of the data because in Section 6.3 the remaining 20% are used as the test set.

Data set	n	$\bar{n}$	l	k
mnist	780	150	60,000	10
rcv1	47,236	65	518,571	53
sector	55,197	163	6,412	105
aloi	128	31	86,400	1,000

We compare the five settings considered in Section 6.1, present results in Figure 6, and have the following observations.

1. The same as in Section 6.1, we still have

$$\textit{Method 2} > \textit{Method 1} > \textit{Subsampled}.$$

   However, the difference between *Method 2* and the other two is generally smaller. This seems to indicate that the direction $\boldsymbol{d}_k$ obtained using subsampled Hessian is good enough and the use of two directions by $\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k$ does not give significant improvements. In this regard, the selection of a good initial step size for line search becomes more crucial, so in Figures 6(c) and 6(d), *Method 1* is much better than *Subsampled*, but it is almost the same as *Method 2*.

2. Except Figure 6(a), all subsampled methods (*Subsampled*, *Method 1*, and *Method 2*) are faster than *Full* or *Full-CG10*. This result is similar to what has been reported

Table 3: Summary of the data sets used for experiments on deep networks. n is the number of features. l is the number of training instances. l_t is the number of testing instances. For the last column, the first * means the number of features and the last * means the number of classes. Note that either the set we obtained is already in the range of $[0, 1]$ or we conduct a feature-wise scaling on the data set.

Data set	n	l	l_t	k	Deep structure
pendigits	16	7,494	3,498	10	*-300-200-100-30-*
usps	256	7,291	2,007	10	*-500-250-100-30-*
mnist	780	60,000	10,000	10	*-1000-500-250-30-*
aloi	128	86,400	21,600	1,000	*-1000-*

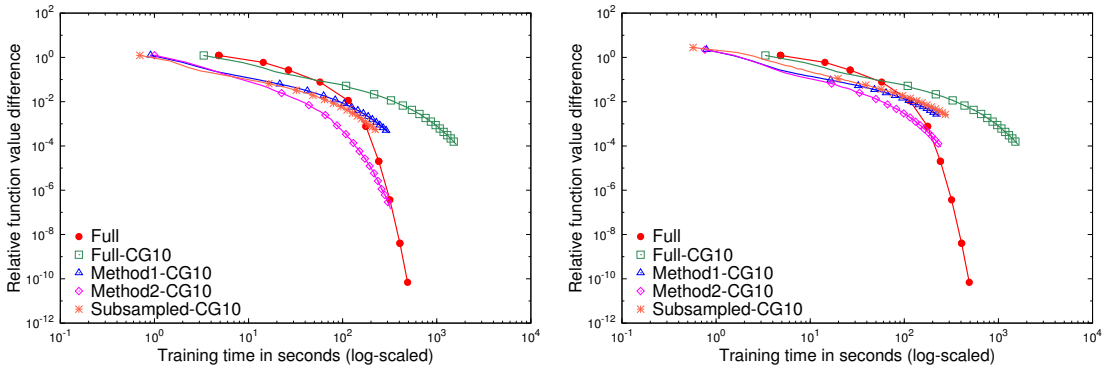
in Byrd et al. [2011]. Therefore, for these multi-class problems, the subsampled Hessian method may have obtained a good enough direction, a situation consistent with our conclusion from the previous observation.

3. *Full* is much faster than *Full-CG10* in Figure 6(a), but is slower in others. This result confirms our earlier finding that a suitable CG_{max} is problem dependent.

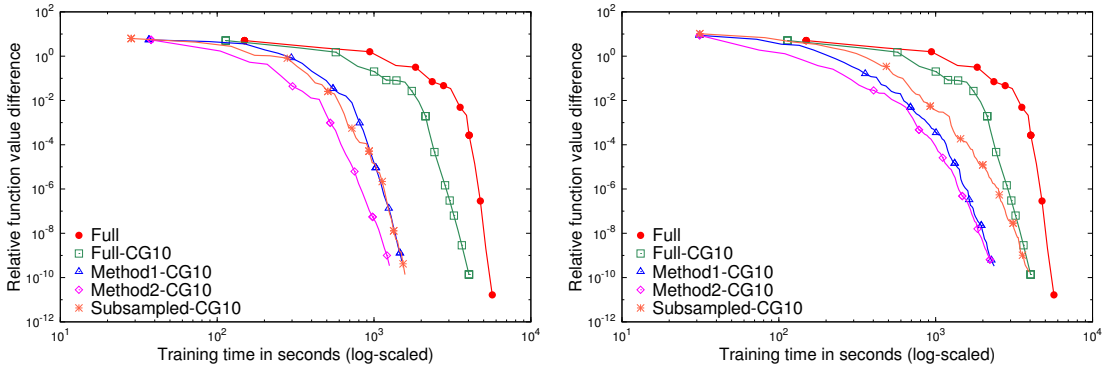
6.3 Deep Learning for Multi-Class Classification

For deep neural networks, we consider some multi-class data sets listed in Table 3. They are different from those used in Section 6.2 because our deep neural network implementation is not suitable for sparse data with many features.

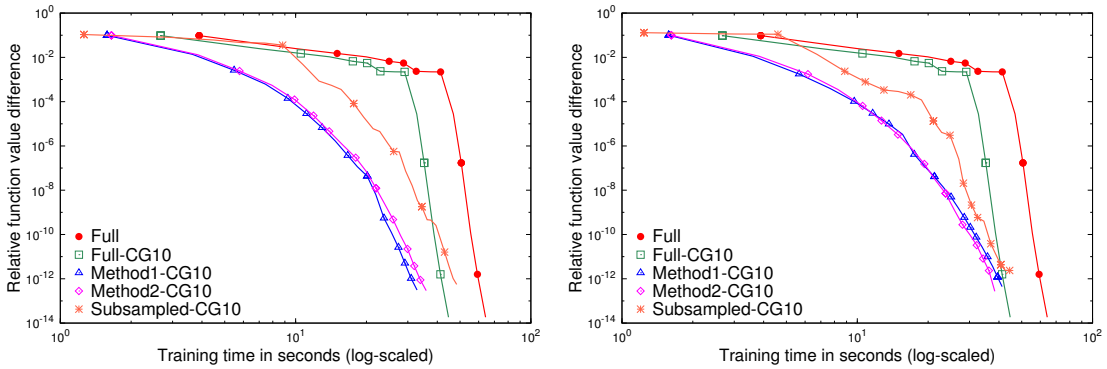
We compare the following methods. The first three are variants of subsampled Hes-



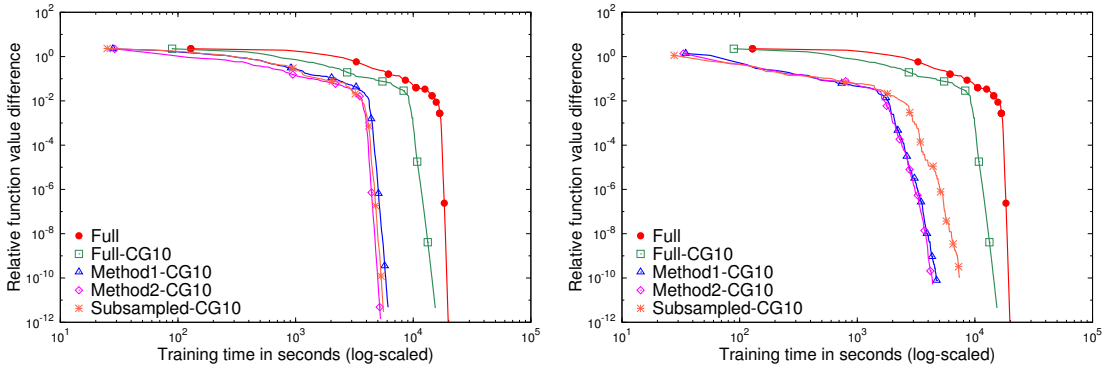
(a) mnist



(b) rcv1



(c) sector



(d) aloi

Figure 6: Experiments on maximum entropy. We present running time (in seconds) versus the relative difference to the optimal function value. Both x-axis and y-axis are log-scaled. Left: $|S_k|/l = 5\%$. Right: $|S_k|/l = 1\%$.

sian methods considered in Martens [2010]. They differ in selecting a CG iterate as the approximate Newton direction. Our experimental framework is modified from the code at `http://www.cs.toronto.edu/~jmartens/docs/HFDemo.zip`. Note that Martens [2010] considers the following stopping condition rather than (6) for the CG procedure. Let $\{\boldsymbol{s}_i\}$ be the sequence of CG iterates. The CG procedure stops at the ith step and let $\boldsymbol{d}_k = \boldsymbol{s}_i$ if

$$i > t \quad \text{and} \quad q_k(\boldsymbol{s}_i) < 0 \quad \text{and} \quad \frac{q_k(\boldsymbol{s}_i) - q_k(\boldsymbol{s}_{i-t})}{q_k(\boldsymbol{s}_i)} < t\epsilon, \tag{41}$$

where $t = \max(10, 0.1i)$ and $\epsilon = 0.0005$.

1. *Martens-sub*: this method, proposed in Martens [2010], stores a subset of CG iterates and selects the one such that the objective value of using the subset S_k (i.e., $\sum_{i \in S_k} \xi(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i)/|S_k|$) is the smallest.

2. *Martens-last*: this method, proposed in Martens [2010], selects the last CG iterate as the search direction. This setting corresponds to the "*Subsampled*" method in Section 6.1.

3. *Martens-full*: this method, proposed in Martens [2010], stores a subset of CG iterates and selects the one such that the objective function value of using the full data set (i.e., $\sum_{i=1}^{l} \xi(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i)/l$) is the smallest.

4. *Comb1*: the method proposed in Section 3 by using the direction $\beta_1 \boldsymbol{d}_k + \beta_2 \bar{\boldsymbol{d}}_k$. Note that (16) is used and all other settings are the same as *Martens-last*. See more details in Section 5.4.

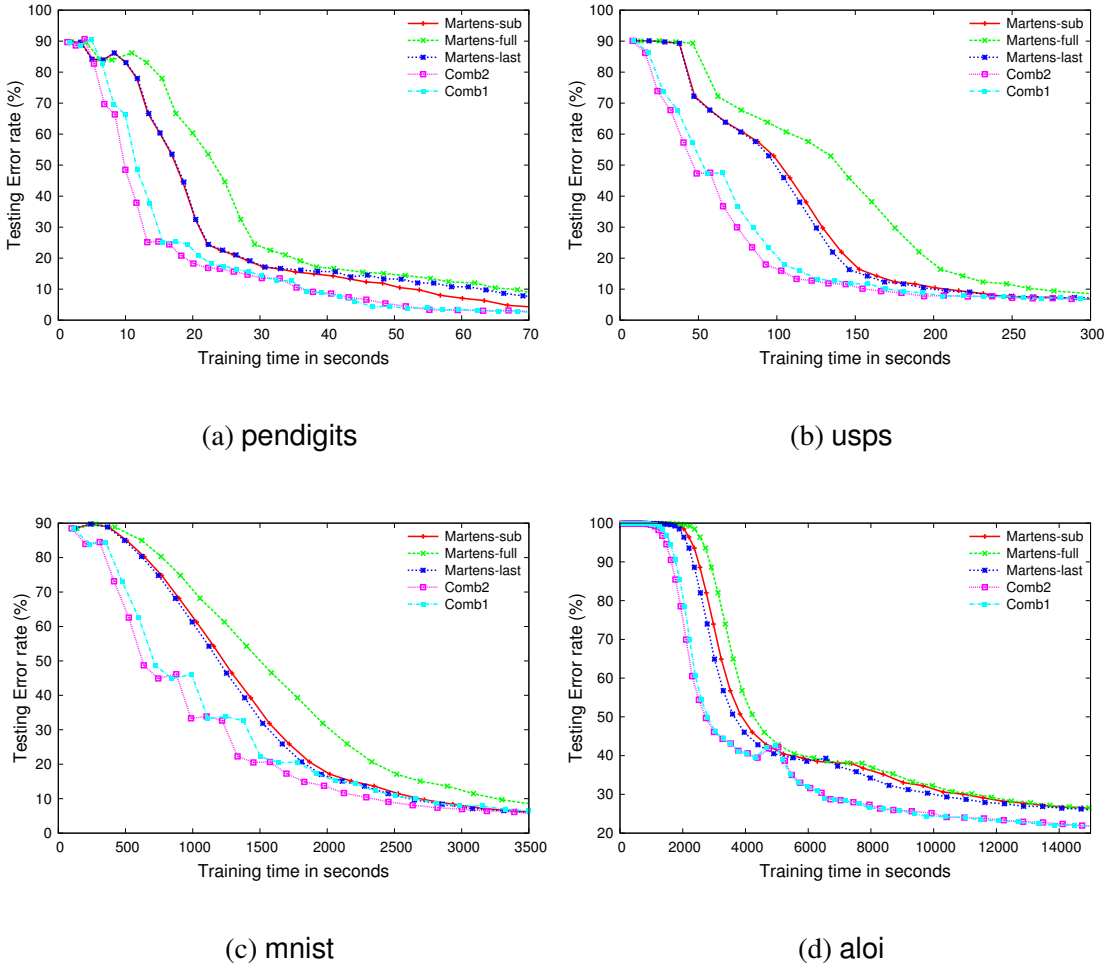
5. *Comb2*: the same as *Comb1* except that the stopping condition (6) is used for the CG procedure.

For the sake of simplicity, we do not implement some tricks used in Martens [2010]:

- No pre-conditioning.
- No use of the previous solution as the initial guess in the CG procedure. The zero vector is used as the initial CG iterate.

For *Comb2*, we set $\sigma = 0.001$ for (6). The value is smaller than 0.1 in Section 6.1 because otherwise the CG stopping condition is too loose. In addition, we consider the same $(\text{boost}, \text{drop}) = (2/3, 3/2)$ as Martens [2010] for the Levenberg-Marquardt method. We use the subsampling rate $|S_k|/l = 5\%$ and $C = 5 \times 10^4$ from Martens [2010] for (36).

From results presented in Figure 7, we observe that the proposed *Comb1* and *Comb2* methods are faster than other methods. Therefore, the optimization problem (16) is useful to improve the quality of the search direction. The difference between *Comb1* and *Comb2* is generally small, but for usps and mnist, *Comb2* is slightly better because of a smaller number of CG iterations per outer iteration. We observe that (6) with $\sigma = 0.001$ is generally looser than (41). Earlier we mentioned that (6) with $\sigma = 0.1$ is too loose. Therefore, with a similar issue discussed earlier on finding suitable CG_{max}, we clearly see the importance of using a CG stopping condition that is neither too loose nor too strict.



(a) pendigits (b) usps

(c) mnist (d) aloi

Figure 7: Experiments on deep neural networks. We present running time (in seconds) versus test error.

7 Conclusions

In this paper, we have proposed novel techniques to improve the subsampled Hessian Newton method. We demonstrate the effectiveness of our method on logistic regression, linear SVM, maximum entropy, and deep neural networks. The asymptotic convergence is proved, and the running time is shown to be shorter than Byrd et al. [2011] and Martens [2010]. This work gives a compelling example of showing that little extra cost in finding the search direction may lead to dramatic overall improvement.

Acknowledgement

This work was supported in part by the National Science Council of Taiwan via the grant 101-2221-E-002-199-MY3.

References

R. H. Byrd, G. M. Chin, W. Neveitt, and J. Nocedal. On the use of stochastic Hessian information in optimization methods for machine learning. *SIAM Journal on Optimization*, 21(3):977–995, 2011.

O. Chapelle and D. Erhan. Improved preconditioner for Hessian free optimization. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-region Methods*. Society for Industrial and Applied Mathematics, Philadelphia, 2000.

Y. Drori and M. Teboulle. Performance of first-order methods for smooth convex minimization: a novel approach. *Mathematical Programming*, 145:451–482, 2014.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: a library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf`.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

I. Griva, S. G. Nash, and A. Sofer. *Linear and nonlinear optimization*. SIAM, second edition, 2009.

F.-L. Huang, C.-J. Hsieh, K.-W. Chang, and C.-J. Lin. Iterative scaling and coordinate descent methods for maximum entropy. *Journal of Machine Learning Research*, 11: 815–848, 2010. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/maxent_journal.pdf`.

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

C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf`.

O. L. Mangasarian. A finite Newton method for classification. *Optimization Methods and Software*, 17(5):913–929, 2002.

O. L. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research*, 7:1517–1530, 2006.

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

J. Martens and I. Sutskever. Training deep and recurrent networks with Hessian-free optimization. In *Neural Networks: Tricks of the Trade*, pages 479–535. Springer, 2012.

J. J. Moré. The Levenberg-Marquardt algorithm: Implementation and theory. In G. Watson, editor, *Numerical Analysis*, number 630 in Lecture Notes in Mathemattcs, pages 105–116, New York, 1978. Sprmger-Verlag.

B. A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural computation*, 6(1): 147–160, 1994.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

N. N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

R. E. Wengert. A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):463–464, 1964.