Sample size selection in optimization methods for machine learning — Source link

Richard H. Byrd, Gillian M. Chin, Jorge Nocedal, Yuchen Wu

Institutions: University of Colorado Boulder, Northwestern University, Google

Published on: 01 Aug 2012 - Mathematical Programming (Springer-Verlag)

Topics: Gradient method, Hessian matrix, Newton's method, Sample size determination and Numerical analysis

Sample Size Selection in Optimization Methods for Machine Learning

Richard H. Byrd[*] Gillian M. Chin[†] Jorge Nocedal [‡] Yuchen Wu[§]

January 16, 2012

Abstract

This paper presents a methodology for using varying sample sizes in batch-type optimization methods for large scale machine learning problems. The first part of the paper deals with the delicate issue of dynamic sample selection in the evaluation of the function and gradient. We propose a criterion for increasing the sample size based on variance estimates obtained during the computation of a batch gradient. We establish an $O(1/\epsilon)$ complexity bound on the total cost of a gradient method. The second part of the paper describes a practical Newton method that uses a smaller sample to compute Hessian vector-products than to evaluate the function and the gradient, and that also employs a dynamic sampling technique. The focus of the paper shifts in the third part of the paper to L_1 regularized problems designed to produce sparse solutions. We propose a Newton-like method that consists of two phases: a (minimalistic) gradient projection phase that identifies zero variables, and subspace phase that applies a subsampled Hessian Newton iteration in the free variables. Numerical tests on speech recognition problems illustrate the performance of the algorithms.

[*]Department of Computer Science, University of Colorado, Boulder, CO, USA. This author was supported by National Science Foundation grant CMMI 0728190 and Department of Energy grant DE-SC0001774.

[†]Department of Industrial Engineering and Management Sciences, Northwestern University. This author was supported by an NSERC fellowship and a grant from Google Inc.

[‡]Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA. This author was supported by National Science Foundation grant DMS-0810213, the ISEN Institute at Northwestern, and by Department of Energy grant DE-FG02-87ER25047-A004.

[§]Google Inc, Mountain View, CA, USA.

1 Introduction

In optimization for supervised machine learning, there exist two regimes in which popular algorithms tend to operate: the stochastic approximation regime, which samples a small data set per iteration, typically a single data point, and the batch or sample average approximation regime, in which larger samples are used to compute an approximate gradient. The choice between these two extremes outlines the well-known tradeoff between inexpensive noisy steps and expensive but more reliable steps, and has given rise to a myriad of optimization algorithms for machine learning and stochastic programming. The availability of parallel computing environments however, has developed a fairly different tradeoff, namely the value of inexpensive sequential steps of stochastic gradient methods versus that of batch steps in which function and gradients are computed in parallel.

In the first part of this paper, we consider the delicate issue of sample selection in the evaluation of function and gradients. We present a practical algorithm for unconstrained stochastic optimization that operates in a mini-batch framework, and that dynamically increases the size of the training sample during the course of the algorithm's progression. The decision to increase the size of the training sample is based on sample variance estimates of batch gradients. By initially selecting a small sample and gradually increasing its size, the algorithm is able to keep overall costs low, while still managing to obtain the desired level of accuracy. In this paper, we study the theoretical properties of a simplified steepest decent method that incorporates this dynamic sample size technique, and we present results pertaining to convergence in expectation, as well as complexity bounds for the total cost of the algorithm.

Sample selection also plays a crucial role in the incorporation of curvature information in a Hessian-free Newton method for machine learning [7, 21]. In this so-called subsampled Hessian Newton-CG method, the step computation is obtained by applying the conjugate gradient (CG) method, which only requires Hessian-vector products and not the Hessian matrix itself. Within this context, the central question is how to select a useful training sample for computing Hessian-vector products that is significantly smaller than the sample used for function and gradient computations. In the second part of the paper, we discuss a Newton-CG method that uses dynamic sampling for the evaluation of the function and gradient as well as for the incorporation of Hessian information. The computational advantages of the proposed dynamic method, over more traditional techniques, are illustrated on a speech recognition problem provided by Google.

The third part of the paper considers optimization problems in which the objective function is the sum of a convex loss function and an L_1 regularization term. We propose a two-phase method consisting of an active-set prediction phase and a subspace phase. A minimum norm subgradient is employed to predict the zero components of the solution, and a subsampled Hessian Newton-CG method is applied in the subspace phase. We compare the performance of the algorithm with that of the orthant-wise L-BFGS method [2] on a speech recognition problem, and observe that the subspace phase plays an important role in generating sparse solutions quickly.

This paper can be seen as a continuation of the work in [7], which dealt with Hessian sampling techniques for a Newton-CG method. Here we take this work further, first by considering also sample selection techniques for the evaluation of function and gradients, and second, by studying the extension of Hessian sampling techniques to nonsmooth L_1 regularized problems.

2 Preliminaries

The optimization problems analyzed in this paper have their origin in large-scale machine learning, and with appropriate modifications, are also relevant to a variety of stochastic optimization applications.

Let $\mathcal{X} \times \mathcal{Y}$ denote the space of input output pairs (x, y) endowed with a probability distribution $P(x, y)$. The objective is to determine the values of the parameters $w \in \mathbb{R}^m$ of a prediction function $f(w; x)$, which we assume to be linear in w. That is, given a training point $x \in \mathbb{R}^m$, we have that

$$f(w; x) = w^T x.$$

The discrepancy between the predicted value $\hat{y} \overset{\text{def}}{=} f(w; x)$ and the known output y, for a given input x, is measured by a convex loss function $l(\hat{y}, y)$. Ideally, the optimization algorithm computes the parameters w of the prediction function $f(w; x)$ so as to minimize the objective

$$\int l(f(w; x), y) \partial P(x, y). \tag{2.1}$$

However, since the distribution $P(x, y)$ is unknown, we use the data set that is at our disposal and minimize instead the convex empirical loss function

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} l(f(w; x_i), y_i), \tag{2.2}$$

which is defined over the entire data set $\{(x_i, y_i)\}_{i=1,\ldots,N}$. (In section 6, we consider the case when the objective (2.2) also contains a regularization term of the form $\|w\|_1$.) We assume that the size the data set, indicated by N, is extremely large, numbered somewhere in the millions or billions, so that the evaluation of $J(w)$ is very expensive.

Stochastic gradient algorithms, such as the Robbins-Monro method [26], its averaging variant proposed by Polyak and Juditsky [25], and Nesterov's dual averaging method [22], have both low computational costs and fast learning rates, and have found a great deal of success in solving large scale machine learning problems where the sheer volume of data makes a full batch approach prohibitively expensive. These methods must, however, perform a vast number of iterations before an appreciable improvement in the objective is obtained, and due to the sequential nature of these iterations, it can be difficult to parallelize them; see [23, 1, 10] and the references therein. On the other hand, batch (or mini-batch) algorithms can easily exploit parallelism in the function and gradient evaluation, and are able to yield high accuracy in the solution of the optimization problem [30, 2, 31], if so desired. Motivated by the potential of function/gradient parallelism, the sole focus of this paper is on batch and mini-batch methods.

3 The Dynamic Sample Size Gradient Method

In this section, we describe a gradient-based mini-batch optimization algorithm in which the size of the training sample used in the evaluation of the function and gradient increases throughout the progression of the algorithm. At every iteration, the method chooses a subset $\mathcal{S} \subset \{1., ..., N\}$ of the training set, and applies one step of an optimization algorithm to the objective function

$$J_{\mathcal{S}}(w) = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} l(f(w; x_i), y_i). \tag{3.1}$$

At the start, a relatively small sample size $|\mathcal{S}|$ is chosen, and if it is judged that the optimization step is likely to produce improvement in the "target" objective function J given in (2.2), the sample *size* is kept unchanged, a new sample is selected, and a new optimization step is computed. Otherwise, the algorithm chooses a larger sample size, selects a corresponding sample $\mathcal{S}$, and computes the next step. The benefit of this approach is that the use of a small initial sample allows rapid progress in the early stages, while a larger sample yields high accuracy in the solution, if so desired. The two crucial ingredients in this sampling strategy are the condition that triggers an increase in the sample size and the rule for choosing the new sample. We must therefore devise a measure of the quality of the sample $\mathcal{S}$, and for this purpose, we use the variance in the gradient $\nabla J_{\mathcal{S}}$.

To motivate our approach, we note that the optimization algorithm, which enforces descent in $J_{\mathcal{S}}$ at every iteration, must also make progress on the target objective function J *sufficiently often* to achieve convergence to the solution. It is easy to verify that the vector $d = -\nabla J_{\mathcal{S}}(w)$ is a descent direction for J at w if

$$\delta_{\mathcal{S}}(w) \overset{\text{def}}{=} \|\nabla J_{\mathcal{S}}(w) - \nabla J(w)\|_2 \leq \theta \|\nabla J_{\mathcal{S}}(w)\|_2, \qquad \text{where } \theta \in [0, 1). \tag{3.2}$$

The quantity $\delta_{\mathcal{S}}(w)$ is, however, not available, since the evaluation of ∇J is prohibitively expensive in applications involving very large training sets. Therefore, we propose approximating $\delta_{\mathcal{S}}(w)$ by an estimate of the variance of the random vector $\nabla J_{\mathcal{S}}(w)$. This will allow us to select the sample $\mathcal{S}$ so that the inequality (3.2) holds sufficiently often.

Before proceeding, it is convenient to simplify the notation. Let us write the prediction-loss function for a given data point i, with respect to the variable w, as

$$\ell(w; i) \overset{\text{def}}{=} l(f(w; x_i), y_i).$$

As a result, we have

$$J_{\mathcal{S}}(w) = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \ell(w; i) \quad \text{and} \quad \nabla J_{\mathcal{S}}(w) = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla \ell(w; i). \tag{3.3}$$

For a given $w \in \mathbb{R}^m$, the expected value of $\nabla J_{\mathcal{S}}(w)$ over all possible samples $\mathcal{S} \subset \{1, \ldots, N\}$ equals $\nabla J(w)$, so that the quantity $\delta_{\mathcal{S}}(w)$ we wish to estimate satisfies

$$\mathbb{E}[\delta_{\mathcal{S}}(w)^2] = \mathbb{E}[\|\nabla J_{\mathcal{S}}(w) - \nabla J(w)\|_2^2] = \|\text{Var}(\nabla J_{\mathcal{S}}(w))\|_1, \tag{3.4}$$

where $\mathrm{Var}(\nabla J_{\mathcal{S}}(w))$ is a vector of the same length as w. Therefore we need to estimate the variance of $\nabla J_{\mathcal{S}}(w)$, where $\mathcal{S}$ is a sample chosen without replacement. Following [16, p.183], we have

$$\mathrm{Var}(\nabla J_{\mathcal{S}}(w)) = \frac{\mathrm{Var}(\nabla \ell(w;i))}{|\mathcal{S}|} \frac{(N - |\mathcal{S}|)}{N-1}. \tag{3.5}$$

Since the population variance $\mathrm{Var}(\nabla \ell(w;i))$ is too expensive to compute, we approximate it with the sample variance,

$$\mathrm{Var}_{i\in\mathcal{S}}(\nabla \ell(w;i)) = \frac{1}{|\mathcal{S}|-1} \sum_{i\in\mathcal{S}} \left(\nabla \ell(w;i) - \nabla J_{\mathcal{S}}(w)\right)^2, \tag{3.6}$$

where the square is taken component wise. In other words, by (3.4) and (3.5) we are making the approximation

$$\mathbb{E}[\delta_{\mathcal{S}}(w)^2] \approx \frac{\|\mathrm{Var}_{i\in\mathcal{S}}(\nabla \ell(w;i))\|_1}{|\mathcal{S}|} \frac{(N - |\mathcal{S}|)}{N-1}. \tag{3.7}$$

Therefore, we replace the deterministic condition (3.2) by the inequality

$$\frac{\|\mathrm{Var}_{i\in\mathcal{S}}(\nabla \ell(w;i))\|_1}{|\mathcal{S}|} \frac{(N - |\mathcal{S}|)}{N-1} \le \theta^2 \|\nabla J_{\mathcal{S}}(w)\|_2^2. \tag{3.8}$$

Since we are interested in the case when the training set is very large, we take the limit as $N \to \infty$ and replace (3.8) with the stronger condition

$$\frac{\|\mathrm{Var}_{i\in\mathcal{S}}(\nabla \ell(w;i))\|_1}{|\mathcal{S}|} \le \theta^2 \|\nabla J_{\mathcal{S}}(w)\|_2^2, \tag{3.9}$$

which also corresponds to the case of sampling with replacement.

In our algorithm, if inequality (3.9) is not satisfied, we increase the sample to a size that we predict will satisfy the descent condition. The heuristic for doing so is as follows. Suppose we wish to find a larger sample $\widehat{\mathcal{S}}$, and let us assume that change in sample size is gradual enough that, for any given value of w,

$$\|\mathrm{Var}_{i\in\widehat{\mathcal{S}}}(\nabla \ell(w;i))\|_1 \simeq \|\mathrm{Var}_{i\in\mathcal{S}}(\nabla \ell(w;i))\|_1 \quad \text{and} \quad \|\nabla J_{\widehat{\mathcal{S}}}(w)\|_2 \simeq \|\nabla J_{\mathcal{S}}(w)\|_2. \tag{3.10}$$

Under these simplifying assumptions, we see that condition (3.9) is satisfied if we choose $|\widehat{\mathcal{S}}|$ so that

$$\frac{1}{|\widehat{\mathcal{S}}|} \|\mathrm{Var}_{i\in\mathcal{S}}(\nabla \ell(w;i))\|_1 \le \theta^2 \|\nabla J_{\mathcal{S}}(w)\|_2^2. \tag{3.11}$$

Therefore, in the algorithm we select the new sample by means of the rule

$$|\widehat{\mathcal{S}}| = \frac{\|\mathrm{Var}_{i\in\mathcal{S}}(\nabla \ell(w;i))\|_1}{\theta^2 \|\nabla J_{\mathcal{S}}(w)\|_2^2}. \tag{3.12}$$

We can now describe the algorithm for minimizing the convex objective function J given in (2.2).

```
Algorithm 3.1: Dynamic Sample Gradient Algorithm
Initialize: Choose an initial iterate $w_0$, an initial sample $\mathcal{S}_0$, and a constant $\theta \in (0,1)$.
    Set $k \leftarrow 0$. Evaluate $J_{\mathcal{S}_k}(w_k)$ and $\nabla J_{\mathcal{S}_k}(w_k)$.

Repeat until a convergence test is satisfied:

  1 Compute $d_k = -\nabla J_{\mathcal{S}_k}(w_k)$
  2 Line Search: compute step length $\alpha_k > 0$ such that
        $J_{\mathcal{S}_k}(w_k + \alpha_k d_k) < J_{\mathcal{S}_k}(w_k)$
  3 Define a new iterate:  $w_{k+1} = w_k + \alpha_k d_k$.
  4 Set $k \leftarrow k + 1$.
  5 Choose a sample $\mathcal{S}_k$ such that $|\mathcal{S}_k| = |\mathcal{S}_{k-1}|$.
  6 Compute the sample variance defined in (3.6).
  7 If condition (3.9) is not satisfied, augment $\mathcal{S}_k$ using formula (3.12).
```

The description of the line search in step 2 is deliberately vague to allow for the fixed steplength strategy used in the next section, as well as the line search procedure of section 5 that enforces a sufficient decrease condition. To highlight the theoretical advantages of Algorithm 3.1, in the next section we present a complexity analysis for a gradient-based dynamic sample size algorithm with fixed step length, under both deterministic and stochastic conditions.

4 Complexity of a Dynamic Batch Gradient Method

In many optimization methods, the cost of each iteration is the same, and it is therefore appropriate to establish complexity results that estimate the maximum number of iterations needed to obtain an ϵ-accurate solution. In the dynamic sample size methods discussed in this paper, however, the cost of the iteration varies greatly during the progression of the algorithm. Therefore, in this context it is appropriate to measure the *total amount of work* required to obtain an ϵ-accurate solution.

Bottou and Bousquet [6] give complexity estimates of this kind for steepest descent and Newton methods, both in the deterministic and stochastic settings. They argue that, when accounting for both the estimation and the optimization errors, the most rapidly convergent optimization algorithms are not the best learning algorithms. Specifically, they note that the complexity bound on the total computational cost for the stochastic gradient descent method of Robbins-Monro [26] is generally better than for the batch steepest descent and Newton methods, as well as for an idealized stochastic Newton method. The overall conclusion of [6] is that in the context of large-scale learning, the stochastic gradient descent method is to be preferred over the other methods they consider.

The analysis of Bottou and Bousquet does not, however, cover dynamic batch algorithms such as the one described in Algorithm 3.1. In this section, we study the complexity of a gradient method that employs the described variable sample size strategy and show that its complexity estimate is as good as that of the stochastic gradient descent method. The savings accrued by increasing the sample size in an appropriate manner lead to measurable improvements in the complexity of the algorithm, compared to the batch gradient method studied in [6].

Our analysis is organized in two parts. First, we assume that the sample size is large enough to ensure that the algorithm produces descent directions for the target objective J at every iteration. In practice, this would require that ∇J be computed at every iterate, which is prohibitively expensive. Therefore, in the second part of this section we extend the analysis to the case when ∇J is never computed and the algorithm produces descent directions with high probability.

4.1 Complexity in the Deterministic Case

We consider the minimization of a (deterministic) convex function $J : \mathbb{R}^m \to \mathbb{R}$ by means of the following fixed-steplength steepest descent algorithm

$$w_{k+1} = w_k - \alpha g_k, \tag{4.1}$$

where α is a constant steplength, and g_k is an approximation to $\nabla J(w_k)$. As is common in complexity studies, we consider an algorithm with a fixed steplength α instead of one that performs an Armijo line search because the latter is more difficult to analyze. The goal of this section is to give conditions on g_k that guarantee convergence of the algorithm, and provide a complexity bound for algorithm (4.1). Our main interest in this analysis is to provide motivation for the approach presented in section 4.2.

We assume that J is twice continuously differentiable and uniformly convex, i.e., there exist constants $0 < \lambda < L$ such that

$$\lambda \|d\|_2^2 \le d^T \nabla^2 J(w) d \le L \|d\|_2^2, \quad \text{for all } w. \tag{4.2}$$

Thus, J has a unique minimizer that we denote by w_*.

Since $\nabla J(w_*) = 0$, we have that

$$\nabla J(w_k) = H_k(w_k - w_*), \quad \text{where} \quad H_k = \int_0^1 \nabla^2 J((1-\tau)w_* + \tau w_k) \mathrm{d}\tau. \tag{4.3}$$

Combining this relation with (4.2) we have

$$\begin{aligned} \nabla J(w_k)^T \nabla J(w_k) &= (w_k - w_*)^T H_k^2 (w_k - w_*) \\ &\ge \lambda (w_k - w_*)^T H_k (w_k - w_*). \end{aligned} \tag{4.4}$$

By convexity of J and (4.3) we have

$$\begin{aligned} J(w_*) &\geq J(w_k) + \nabla J(w_k)^T(w_* - w_k) \\ &= J(w_k) + (w_k - w_*)^T H_k (w_* - w_k), \end{aligned}$$

and hence

$$(w_k - w_*)^T H_k (w_k - w_*) \geq J(w_k) - J(w_*).$$

Substituting this in (4.4) we obtain

$$\nabla J(w_k)^T \nabla J(w_k) \geq \lambda[J(w_k) - J(w_*)]. \tag{4.5}$$

Also, note that by (4.2),

$$\begin{aligned} J(w_k) &\geq J(w_*) + \tfrac{\lambda}{2}\|w_k - w_*\|_2^2 \\ &\geq J(w_*) + \tfrac{\lambda}{2L^2}(w_k - w_*)^T H_k^2 (w_* - w_k), \end{aligned}$$

and thus

$$J(w_k) - J(w_*) \geq \tfrac{\lambda}{2L^2}\|\nabla J(w_k)\|_2^2. \tag{4.6}$$

We are now ready to analyze iteration (4.1). For convenience, we assume that $J(w_*) = 0$.

Theorem 4.1 *Suppose that J is twice differentiable and uniformly convex, satisfying* (4.2), *and that $J(w_*) = 0$. Let $\{w_k\}$ be the sequence of iterates generated by iteration* (4.1) *with*

$$\alpha = (1-\theta)/L \qquad \text{and} \qquad \theta \in (0,1). \tag{4.7}$$

If at an iteration k the approximate gradient g_k satisfies

$$\|g_k - \nabla J(w_k)\| \leq \theta \|g_k\|, \tag{4.8}$$

then

$$J(w_{k+1}) \leq \left(1 - \frac{\beta\lambda}{L}\right) J(w_k) \quad \text{with} \quad \beta = \frac{(1-\theta)^2}{2(1+\theta)^2}. \tag{4.9}$$

Moreover, if condition (4.8) *holds at every iteration we have that*

$$\{w_k\} \to w_*, \tag{4.10}$$

and the number of iterations, j, required to obtain $J(w_j) \leq J(w_*) + \epsilon$ is at most

$$\frac{L}{\beta\lambda}[\log(1/\epsilon) + \log(J(w_0))]. \tag{4.11}$$

In addition,

$$\|g_k\|^2 \leq \frac{1}{(1-\theta)^2} \frac{2L^2 J(w_0)}{\lambda} \left(1 - \frac{\beta\lambda}{L}\right)^k. \tag{4.12}$$

Proof. Suppose that condition (4.8) holds at iteration k. Then we have both

$$\|\nabla J(w_k)\| \leq (1+\theta)\|g_k\| \qquad \text{and} \qquad \|\nabla J(w_k)\| \geq (1-\theta)\|g_k\|. \tag{4.13}$$

The second inequality and (4.8) yield

$$\|g_k - \nabla J(w_k)\| \leq \tfrac{\theta}{(1-\theta)}\|\nabla J(w_k)\|.$$

Squaring (4.8) and recalling (4.13) we obtain

$$\begin{aligned} 2\nabla J(w_k)^T g_k &\geq (1-\theta^2)\|g_k\|^2 + \|\nabla J(w_k)\|^2 \\ &\geq [1-\theta^2+(1-\theta)^2]\|g_k\|^2, \end{aligned}$$

and thus

$$\nabla J(w_k)^T g_k \geq (1-\theta)\|g_k\|^2, \tag{4.14}$$

implying that $-g_k$ is a descent direction for the objective function J. It follows from Taylor's theorem (4.1), (4.2), (4.7), (4.13) and (4.14) that

$$\begin{aligned} J(w_{k+1}) &\leq J(w_k) - \tfrac{(1-\theta)}{L}\nabla J(w_k)^T g_k + \tfrac{L}{2}(\tfrac{1-\theta}{L})^2\|g_k\|^2 \qquad (4.15) \\ &\leq J(w_k) - \tfrac{(1-\theta)^2}{L}\|g_k\|^2 + \tfrac{L}{2}(\tfrac{1-\theta}{L})^2\|g_k\|^2 \\ &\leq J(w_k) - \tfrac{(1-\theta)^2}{2L}\|g_k\|^2 \\ &\leq J(w_k) - \tfrac{(1-\theta)^2}{2L(1+\theta)^2}\|\nabla J(w_k)\|^2. \end{aligned}$$

This can be summarized as

$$J(w_{k+1}) \leq J(w_k) - \frac{\beta}{L}\|\nabla J(w_k)\|^2, \tag{4.16}$$

where β is given in (4.9). Combining this relation with (4.5) we obtain

$$J(w_{k+1}) \leq J(w_k) - \tfrac{\beta\lambda}{L}(J(w_k) - J(w_*)),$$

and since we assume that $J(w_*) = 0$, we have

$$J(w_{k+1}) \leq (1 - \tfrac{\beta\lambda}{L})J(w_k),$$

which proves (4.9).

Suppose now that condition (4.8) holds at every iteration. Then, we have from (4.9) that

$$J(w_k) \leq (1 - \tfrac{\beta\lambda}{L})^k J(w_0), \tag{4.17}$$

and since the term inside the brackets is less than 1, we have that $J(w_k) \rightarrow J(w_*) = 0$. Furthermore, since J is uniformly convex, this implies (4.10).

To establish (4.12), we combine (4.17), (4.6), (4.13) and the assumption that $J(w_*) = 0$.

In order to obtain a complexity estimate, we need to bound the percentage of decrease in J at every iteration. It follows from (4.17) that

$$\begin{aligned} \log J(w_k) &\leq k\log(1-\tfrac{\beta\lambda}{L}) + \log J(w_0) \\ &\leq -k\tfrac{\beta\lambda}{L} + \log J(w_0), \end{aligned}$$

where we used the bound $\log(1-x) \leq -x$, which follows by concavity of the log function. Thus, for all k we have that

$$k \leq \tfrac{L}{\beta\lambda}[\log J(w_0) - \log J(w_k)].$$

We now reason indirectly and suppose that $J(w_k) \geq \epsilon$. This implies from the above relation that k satisfies

$$k \leq \tfrac{L}{\beta\lambda}[\log J(w_0) - \log \epsilon].$$

Conversely, if the last relation does not hold, i.e.,

$$k > \tfrac{L}{\beta\lambda}[\log J(w_0) + \log(1/\epsilon)], \tag{4.18}$$

then we must have that $J(w_k) < \epsilon$. In summary, the number of iterations to reduce $J(w)$ below $J(w_*) + \epsilon$ is at most (4.11). $\square$

In proving this result, we have assumed that J is uniformly convex, which may be seen as a limitation since some loss functions, such as logistic functions, are only convex. Nevertheless, it is common in practice to add a regularization term of the form $\frac{1}{2}\gamma w^T w$ to the objective (2.2), which guarantees uniform convexity. Moreover, uniform convexity is useful in our setting since it allows us to highlight the effect of the condition number L/λ of the Hessian on the complexity bound of the iteration.

4.2 Stochastic Analysis

We now consider the case when the objective function J is given by the expectation (2.1) and the approximate gradient is defined as

$$g_k = \nabla J_{\mathcal{S}_k}(w_k) = \frac{1}{n_k}\sum_{i\in\mathcal{S}_k} \nabla \ell(w_k; i), \tag{4.19}$$

where from now on n_k denotes the sample size at iteration k, i.e.,

$$n_k \overset{\text{def}}{=} |S_k|.$$

Given the stochastic nature of the problem, we could try to derive conditions of the sample size under which the gradient accuracy condition (4.8) is satisfied in expectation. As discussed in section 3 (see (3.9)), it would then be appropriate to impose the condition

$$\frac{\|\mathrm{Var}_{i\in\mathcal{S}_k}(\nabla \ell(w;i))\|_1}{n_k} \leq \theta^2 \|g_k\|_2^2, \qquad \text{where } \theta \in [0,1). \tag{4.20}$$

However, because of sample variation, it is not possible to guarantee that (4.20) is satisfied at each iteration. Therefore, in our analysis we follow a different approach in which we impose a minimal rate of increase in n_k, rather than basing the choice of the sample size on (4.20). To motivate this strategy, we make the following observations.

By (4.12), satisfaction of condition (4.20) at every iteration leads to

$$\frac{\|\mathrm{Var}_{i\in\mathcal{S}_k}(\nabla\ell(w;i))\|_1}{n_k} \le \frac{\theta^2}{(1-\theta)^2}\frac{2L^2J(w_0)}{\lambda}\left(1-\frac{\beta\lambda}{L}\right)^k. \tag{4.21}$$

Let us assume the existence of a constant ω such that,

$$\|\mathrm{Var}(\nabla\ell(w_k,i))\|_1 \le \omega \qquad \forall\ w_k. \tag{4.22}$$

For the sake of the discussion, let us assume that this bound applies also to the sample variance that appears on the left hand side of (4.21). Then, from that relation we obtain the following bound on the sample size:

$$n_k \ge O\left((1-\tfrac{\beta\lambda}{L})^{-k}\right).$$

Thus the gradient accuracy condition (4.8) leads to linear convergence and suggests that the sample size n_k should grow geometrically with k. In our analysis we will therefore require that the sample size satisfies

$$n_k = a^k \quad \text{for some } a>1, \tag{4.23}$$

This will allow us to show linear convergence in expectation and to estimate the total cost of achieving a specified accuracy.

We consider the dynamic batch steepest descent iteration

$$w_{k+1} = w_k - \tfrac{1}{L}g_k, \tag{4.24}$$

where L is defined in (4.2) and the sample size changes at every iteration according to the rule (4.23). We have omitted the term $(1-\theta)$ used in (4.7) because it has no effect on the expected value analysis presented in this section (i.e., shortening the steplength by the factor $(1-\theta)$ is only needed for the deterministic Theorem 6.1 that imposed (4.8) directly).

We have by Taylor's theorem and (4.2) that

$$J(w_{k+1}) \le J(w_k) - \tfrac{1}{L}\nabla J(w_k)^Tg_k + \tfrac{1}{2L}||g_k||^2.$$

If we condition on w_k, the only random quantity in this expression is g_k, which is the sample mean of the gradients $\nabla\ell(w;i)$, for $i\in\mathcal{S}_k$. Taking expectations, and using the equality $\mathbb{E}[g_k]=\nabla J(w_k)$, we have

$$\begin{aligned}\mathbb{E}[J(w_{k+1})] &\le J(w_k) - \tfrac{1}{L}||\nabla J(w_k)||^2 + \tfrac{1}{2L}\mathbb{E}[||g_k||^2]\\ &= J(w_k) - \tfrac{1}{L}||\nabla J(w_k)||^2 + \tfrac{1}{2L}(||\nabla J(w_k)||^2 + \|\mathrm{Var}(g_k)\|_1),\end{aligned} \tag{4.25}$$

where $\text{Var}(g_k)$ is the *true* variance of the sample mean g_k, and we have used the fact that for each component g_k^i of the vector g_k we have that $\text{Var}(g_k^i) = \mathbb{E}[(g_k^i)^2] - (\mathbb{E}[g_k^i])^2$. From (3.5), we note that

$$||\text{Var}(g_k)||_1 = \frac{||\text{Var}(\nabla \ell(w;i))||_1}{n_k} \frac{(N-n_k)}{N-1} \leq \frac{||Var(\nabla \ell(w;i))||_1}{n_k}$$

As a result, the conditional expectation satisfies

$$\mathbb{E}[J(w_{k+1})] \leq J(w_k) - \frac{1}{2L}||\nabla J(w_k)||^2 + \frac{1}{2Ln_k}\|\text{Var}(\nabla \ell(w,i))\|_1. \tag{4.26}$$

Now, using (4.5) and (4.22) we have that, for all k,

$$\mathbb{E}[J(w_{k+1}) - J(w_*)] \leq \left(1 - \frac{\lambda}{2L}\right)(J(w_k) - J(w_*)) + \frac{\omega}{2Ln_k}. \tag{4.27}$$

We can now establish convergence in expectation for the algorithm given by (4.23), (4.24).

Theorem 4.2 *Suppose that for all k the sample size $n_k = |\mathcal{S}_k|$ used in the definition* (4.19) *of g_k satisfies the relation $n_k \geq a^k$ for some $a > 1$, and that there is a constant ω such that* (4.22) *holds for all w_k. Then the sequence $\{w_k\}$ generated by the dynamic batch algorithm* (4.24) *satisfies*

$$\mathbb{E}[J(w_k) - J(w_*)] \leq C\rho^k, \tag{4.28}$$

for all k, where

$$\rho = \max\{1 - \lambda/(4L), 1/a\} < 1 \quad \textit{and} \quad C = \max\{J(w_0) - J(w_*), 2\omega/\lambda\}. \tag{4.29}$$

Proof. Note that since $\mathbb{E}[J(w_0) - J(w_*)] \leq C$, inequality (4.28) holds for $k = 0$. Now, suppose that (4.28) holds for some k. Then by (4.27), the condition $n_k \geq a^k$, the inequality $\rho a > 1$, and (4.29) we have,

$$\begin{aligned}
\mathbb{E}[J(w_{k+1}) - J(w_*)] &\leq \left(1 - \frac{\lambda}{2L}\right) C\rho^k + \frac{\omega}{2Ln_k} \\
&\leq C\rho^k \left(1 - \frac{\lambda}{2L} + \frac{\omega}{2LC(\rho a)^k}\right) \\
&\leq C\rho^k \left(1 - \frac{\lambda}{2L} + \frac{\omega}{2LC}\right) \\
&\leq C\rho^k \left(1 - \frac{\lambda}{2L} + \frac{\lambda}{4L}\right) \\
&\leq C\rho^k \left(1 - \frac{\lambda}{4L}\right) \\
&\leq C\rho^{k+1}.
\end{aligned}$$

□

This result shows that $\mathbb{E}[J(w_k) - J(w_*)] \to 0$ sufficiently rapidly, which allows us to establish a bound on the total computational work required to obtain an ϵ-optimal solution.

Corollary 4.3 *The total number of evaluations of the gradients $\nabla \ell(w;i)$ needed to obtain an ϵ-optimal solution is $O(L/\lambda\epsilon)$, and the total amount of work of algorithm* (4.23)-(4.24) *is*

$$O\left(\frac{Lm}{\lambda\epsilon}\max\{J(w_0)-J(w_*), 2\omega/\lambda\}\right), \tag{4.30}$$

where m is the number of variables.

Proof. From (4.28) we have that the number of iterations k to achieve

$$\mathbb{E}[J(w_k)-J(w_*)] \le \epsilon$$

satisfies

$$C\rho^k \le \epsilon,$$

or equivalently,

$$k\log\rho \le \log\epsilon - \log C.$$

This is satisfied if k is as large as

$$\frac{\log(\epsilon^{-1})+\log C}{\log(\rho^{-1})}. \tag{4.31}$$

To estimate the cost of k steps, we recall that the sample size satisfies $n_k = a^k$, so that at iteration k the computation of the batch gradient g_k requires a^k evaluations of the gradients $\nabla\ell(w_k, i)$. Now, for k given by (4.31), we have

$$\begin{aligned} a^k &= a^{\frac{\log\epsilon^{-1}+\log C}{\log\rho^{-1}}} \\ &= \exp\left(\log(a)\frac{\log\epsilon^{-1}+\log C}{\log\rho^{-1}}\right) \\ &= \left(\exp(\log\epsilon^{-1}+\log C)\right)^{\frac{\log a}{-\log\rho}} \\ &= \left(\frac{C}{\epsilon}\right)^{\tau}, \end{aligned}$$

where $\tau = \frac{\log a}{-\log\rho}$. Therefore, the total number of sample point gradient evaluations for the first k iterations is

$$\sum_{j=0}^{k} a^j = \frac{a^{k+1}-1}{a-1} = \frac{\left(\frac{C}{\epsilon}\right)^{\tau}a-1}{a-1} \le \frac{\left(\frac{C}{\epsilon}\right)^{\tau}}{1-1/a}.$$

Note that by (4.29) we have that $\tau = \frac{\log a}{-\log\rho} \ge 1$. If we choose a so that $a \le (1-\frac{\lambda}{4L})^{-1}$, we have that $\rho = 1/a$, and hence $\tau = 1$. In particular, if we let $a = (1-\sigma\frac{\lambda}{4L})^{-1}$ for some scalar $\sigma \le 1$, then $\tau = 1$ and we have

$$\sum_{j=0}^{k} a^j = \frac{4L}{\sigma\lambda}\frac{C}{\epsilon}.$$

If we assume that the cost of evaluating each gradient $\nabla \ell(w; i)$ is $O(m)$, and by the definition (4.29), we obtain (4.30). $\square$

The complexity result (4.30) holds if the constant a in the rule $n_k \geq a^k$ is chosen in the interval $(1, (1 - \frac{\lambda}{4L})^{-1}]$. We can therefore select an appropriate value of a if the condition number $\kappa = L/\lambda$ is known. But it is not necessary to know this condition number exactly; any overestimate will do. Specifically, the complexity bound will be valid for $a = (1 - \frac{1}{4\kappa'})^{-1}$ where $\kappa' \geq \kappa$.

Let us compare (4.30) with the complexity bounds computed by Bottou and Bousquet [6] for a fixed sample gradient method and a stochastic gradient method, which are described as follows:

$$\text{Fixed Sample Gradient Method:} \qquad w_{k+1} = w_k - \frac{1}{L} \frac{\sum_{i=1}^{N_\epsilon} \nabla \ell(w_k, i)}{N_\epsilon} \tag{4.32}$$

$$\text{Stochastic Gradient Method:} \qquad w_{k+1} = w_k - \frac{1}{\lambda k} \nabla \ell(w_k; i_k), \tag{4.33}$$

where N_ϵ specifies a sample size such that $|J(w_*) - J_{\mathcal{S}}(w_*)| < \epsilon$. The complexity bounds presented in [6] ignore the effect of the initial point w_0, so to make a comparison possible, we also remove the effect of the initial point from the bound (4.30).

Algorithm Name	Bound	Algorithm Description
Dynamic Sample Gradient Method	$O\left(m\kappa\omega/\lambda\epsilon\right)$	(4.23), (4.24)
Fixed Sample Gradient Method	$O(m^2\kappa\epsilon^{-1/\bar{\alpha}} \log^2 \frac{1}{\epsilon})$	(4.32)
Stochastic Gradient Method	$O(m\bar{\nu}\kappa^2/\epsilon)$	(4.33)

Table 4.1: Complexity Bounds for Three Methods. Here m is the number of variables, $\kappa = L/\lambda$ is the condition number, ω and $\bar{\nu}$ are defined in (4.22), (4.34), and $\bar{\alpha} \in [1/2, 1]$.

The scalar $\bar{\alpha} \in [1/2, 1]$, called the estimation rate [6], depends on properties of the loss function; the constant $\bar{\nu}$ is defined as

$$\bar{\nu} = \text{trace}(H^{-1}G), \quad \text{where} \quad H = \nabla^2 J(w_*), \quad G = \mathbb{E}\left[\nabla \ell(w_*; i) \nabla \ell(w_*; i)^T\right]. \tag{4.34}$$

From this definition, we see that $\bar{\nu}$ is roughly comparable to the quantity ω/λ, where ω is given in (4.22). Under this approximation, we have from the third line of Table 4.1 that the bound for the stochastic gradient method is $O\left(m\omega\kappa^2/\lambda\epsilon\right)$, which is very similar to the bound for the dynamic gradient method — the main difference is in the presence of the term κ^2 instead of κ. We thus conclude that the dynamic sample gradient method enjoys a complexity bound that is competitive with the one for the stochastic gradient method.

So far in this paper, the discussion and analysis of the dynamic sampling strategy has focused on pure gradient methods. We can, however, extend this strategy to more powerful optimization methods. This is the subject of the next section.

5 A Newton-CG Method with Dynamic Sampling

As is well known, the steepest descent method can be slow to converge, and algorithms that incorporate curvature information about the objective function can be more efficient. In this section we describe a Hessian-free Newton-CG method that employs dynamic sampling techniques to compute the function and gradient, as well as Hessian-vector products. The focus of this section is on the development of a practical algorithm, not on establishing complexity bounds, and therefore our method is an extension of Algorithm 3.1 and not of the idealized variant given by (4.23)-(4.24).

This paper goes beyond [7] in two respects. In [7] it is assumed that the sample $\mathcal{S}_k$ used to evaluate the function and gradient is fixed, and here we incorporate the dynamic sampling techniques of Algorithm 3.1. In addition, we propose a termination test for the inner CG iteration that is suitable for a dynamic sample size algorithm, whereas in [7] the number of CG iterations is fixed.

At every iteration, the subsampled Newton-CG method chooses a samples $\mathcal{S}_k$ and $\mathcal{H}_k$ such that $|\mathcal{H}_k| \ll |\mathcal{S}_k|$, and defines the search direction d_k as an approximate solution of the linear system

$$\nabla^2 J_{\mathcal{H}_k}(w_k)d = -\nabla J_{\mathcal{S}_k}(w_k). \tag{5.1}$$

The solution is computed by the Hessian-free conjugate gradient method in which $\nabla^2 J_{\mathcal{H}_k}(w_k)$ is never formed; instead the product of this Hessian times a vector is coded directly. The requirement that the sample $\mathcal{H}_k$ be much smaller than the sample $\mathcal{S}_k$ is crucial, for it allows the Hessian-vector product to be inexpensive enough to make the Newton-CG method affordable. In order to avoid having too many parameters to tune, we assume henceforth that the ratio

$$R = |\mathcal{H}_k|/|\mathcal{S}_k| \tag{5.2}$$

is constant throughout the progression of the algorithm. Thus, although the sizes of $\mathcal{S}_k$ and $\mathcal{H}_k$ may change, their ratio remains constant. A guideline in the choice of R is that the *total* cost of Hessian-vector products in the CG algorithm should not be larger than the evaluation of one gradient $\nabla J_{\mathcal{S}_k}$. In other words, the cost of one iteration of the Newton-CG algorithm will be less than twice the cost of a steepest descent step that uses the gradient $\nabla J_{\mathcal{S}_k}$.

5.1 The Conjugate Gradient Iteration

We propose an automatic criterion for deciding the accuracy in the solution of the system (5.1). It is based on the observation that the residual,

$$r_k \overset{\text{def}}{=} \nabla^2 J_{\mathcal{H}_k}(w_k)d + \nabla J_{\mathcal{S}_k}(w_k), \tag{5.3}$$

need not be smaller than the accuracy with which $\nabla^2 J_{\mathcal{H}_k}(w_k)$ approximates $\nabla^2 J_{\mathcal{S}_k}(w_k)$. To be more precise, let us write

$$\nabla^2 J_{\mathcal{S}_k}(w_k)d + \nabla J_{\mathcal{S}_k}(w_k) = r_k + \left[\nabla^2 J_{\mathcal{S}_k}(w_k) - \nabla^2 J_{\mathcal{H}_k}(w_k)\right] d. \tag{5.4}$$

The term on the left hand side is the residual of the standard Newton iteration in which the (larger) sample $\mathcal{S}_k$ is used both for the Hessian and the gradient computations. The last term on the right hand side,

$$\Delta_{\mathcal{H}_k}(w_k; d) \overset{\text{def}}{=} \left[\nabla^2 J_{\mathcal{S}_k}(w_k) - \nabla^2 J_{\mathcal{H}_k}(w_k)\right] d, \tag{5.5}$$

measures the error in the Hessian approximation, along the direction d, due to the use of a smaller sample $\mathcal{H}_k$. It is apparent from (5.4) that it is inefficient to require that the residual r_k be significantly smaller than the Hessian approximation error $\Delta_{\mathcal{H}_k}(w_k; d)$, as the extra effort in solving the linear system may not lead to an improved search direction for the objective function $J_{\mathcal{S}_k}(w)$. In other words, our goal is to balance the two terms on the right hand side of (5.4) and to terminate the CG iteration when the residual r_k is smaller than $\Delta_{\mathcal{H}_k}(w_k; d)$, in norm.

The vector $\Delta_{\mathcal{H}_k}(w; d)$, like $\delta_{\mathcal{S}}(w)$ in (3.2), is, however, not readily available. The CG algorithm effectively computes the product $\nabla^2 J_{\mathcal{H}_k}(w_k)d$ at every iteration, but computing the term $\nabla^2 J_{\mathcal{S}_k}(w_k)d$ would defeat the purpose of the Hessian sub-sampling approach. Therefore, in a similar fashion to the gradient estimation technique of section 3, we use the variance of $\nabla^2 J_{\mathcal{H}_k}(w_k)d$ to approximate (5.5).

Specifically, as in (3.7) we make the approximation

$$\begin{aligned}\mathbb{E}[\Delta_{\mathcal{H}_k}(w_k; d)^2] &\approx \frac{\|\mathrm{Var}_{i\in\mathcal{H}_k}\left(\nabla^2 \ell(w_k; i)d\right)\|_1}{|\mathcal{H}_k|}\frac{(|\mathcal{S}_k| - |\mathcal{H}_k|)}{|\mathcal{S}_k| - 1} \\ &\approx \frac{\|\mathrm{Var}_{i\in\mathcal{H}_k}\left(\nabla^2 \ell(w_k; i)d\right)\|_1}{|\mathcal{H}_k|},\end{aligned} \tag{5.6}$$

where the second approximation follows from the condition $|\mathcal{H}_k| \ll |\mathcal{S}_k|$.

Now, since the vector d is itself generated by the CG iteration, one, in principle, needs to recompute the variance at every CG iteration. This may not be particularly expensive in some settings, but in our implementation we employ a simpler strategy, where the variance calculation is performed only once, at the beginning of the CG algorithm.

If we initialize the CG iteration at the zero vector, we have from (5.3) that the initial CG search direction is given by $p_0 = -r_0 = -\nabla J_{\mathcal{S}_k}(w_k)$. We compute the last term in (5.6) at the beginning of the CG iteration, for $d = p_0$. The stop test for the $j+1$ CG iteration is then set as

$$\|r_{j+1}\|_2^2 \le \Psi \overset{\text{def}}{=} \left(\frac{\|\mathrm{Var}_{i\in\mathcal{H}_k}\left(\nabla^2 \ell(w; i)p_0\right)\|_1}{|\mathcal{H}_k|}\right)\frac{\|d_j\|_2^2}{\|p_0\|_2^2},$$

where d_j is the jth trial candidate for the solution of (5.1) generated by the CG process and the last ratio accounts for the the length of the CG solution. We describe this method in Algorithm 5.1.

Algorithm 5.1: Modified Conjugate Gradient Method for Problem (5.1)

```
Set $d_0 = 0$, $\Psi = 0$. Compute $r_0 = \nabla J_{\mathcal{S}_k}(w_k)$, set $p_0 = -r_0$, $j = 0$
    **While** Residual Convergence Test has not been met, i.e. $||r_j||_2^2 > \Psi$
        1. Compute $s_j = \nabla^2 J_{\mathcal{H}_k}(w_k) p_k$
                If $j = 0$,    compute    $\gamma = \frac{\|\mathrm{Var}_{i \in \mathcal{H}_k}(\nabla^2 \ell(w;i) p_0)\|_1}{|\mathcal{H}_k| \|p_0\|_2^2}$.
        2. Perform CG iteration
            $\alpha_j = r_j^T r_j / p_j^T s_j$
            $d_{j+1} = d_j + \alpha_j p_j$
            $r_{j+1} = r_j + \alpha_j s_j$
            $\beta_{j+1} = r_{j+1}^T r_{j+1} / r_j^T r_j$
            $p_{j+1} = -r_{j+1} + \beta_{j+1} p_j$
            $j = j + 1$
        3. Update stopping tolerance:
            $\Psi = \gamma ||d_j||_2^2$
        **End**
```

We note that, in Algorithm 5.1, we have scaled the tolerance Ψ by the length of the solution vector computed by CG, thereby incorporating some aspect of its scale into the stop test. An additional benefit of this approach is that if the sample size is quite small, and the Hessian matrix for the sample $\mathcal{H}_k$ is nearly singular, then the revised CG algorithm will terminate before the direction vector becomes too long.

5.2 Specification of the Newton-CG Algorithm

We can now describe the subsampled Hessian Newton method for minimizing the target objection function (2.2), that employs the dynamic sample strategy discussed in section 3.

```
Algorithm 5.2: Newton-CG Method with Dynamic Sampling
Initialize: Choose an initial iterate $w_0$, initial samples $\mathcal{H}_0 \subseteq \mathcal{S}_0$, and a sampling ratio
$R$ such that $|\mathcal{H}_0| = R|\mathcal{S}_0|$. Choose constants $\theta \in (0,1)$, $0 < c_1 < c_2 < 1$. Set $k \leftarrow 0$.

Repeat until a convergence test is satisfied:
  1 Compute the search direction $d_k$ by means of Algorithm 5.1.
  2 Compute a step length $\alpha_k$ that satisfies the Wolfe conditions:
      1.  $J_{\mathcal{S}_k}(w_k + \alpha_k d_k) \le J_{\mathcal{S}_k}(w_k) + c_1 \alpha_k \nabla J_{\mathcal{S}_k}(w_k)^T d_k$
      2.  $\nabla J_{\mathcal{S}_k}(w_k + \alpha_k d_k)^T d_k \ge c_2 \nabla J_{\mathcal{S}_k}(w_k)^T d_k$.
  3 Define the new iterate: $w_{k+1} \leftarrow w_k + \alpha_k d_k$.
  4 Increment the counter $k \leftarrow k + 1$.
  5 Choose a sample $\mathcal{S}_k$ such that $|\mathcal{S}_k| = |\mathcal{S}_{k-1}|$.
  6 If condition (3.9) is not satisfied, augment $\mathcal{S}_k$ using formula (3.12).
  7 Select a sample $\mathcal{H}_k \subseteq \mathcal{S}_k$, such that $|\mathcal{H}_k| = R|\mathcal{S}_k|$ .
```

5.3 Numerical Results

To illustrate the benefits of dynamic sampling, we report results of Algorithm 5.2 on the supervised learning speech recognition problem described in [7], which was modeled using a multi-class logistic function. The objective function J is represented as a negative log-likelihood function of the form

$$J(w) = -\frac{1}{N} \sum_{h=1}^{N} \log \frac{\exp(w_{y_h}^T x_h)}{\sum_{i \in \mathcal{C}} \exp(w_i^T x_h)} \tag{5.7}$$

The goal in this speech recognition problem is to determine the parameter w that maximizes the probability of correct classification over the training set, and this is achieved by minimizing (5.7). In this expression, $\mathcal{C} = \{1, 2, \cdots, 128, 129\}$ denotes the set of all class labels; $y_h \in \mathcal{C}$ is the class label associated with data point h; x_h is the feature vector for data point h; and w is a parameter vector of dimension $|\mathcal{C}| \times |F| = 10,191$, where $|F| = 79$ is the number of feature measurements per data point. In (5.7) w_i denotes the parameter sub-vector for class label i. The number of training points is $N = 168776$, and the number of parameters, is $m = 10191$. The training set was provided by Google.

This problem is small enough to be run on a workstation, but sufficiently complex to be representative of production-scale speech recognition problems. We tested Algorithm 5.2 against a variant that we call Algorithm 5.2-Static, in which $|\mathcal{S}_k|$ is constant for all k, i.e., the same sample size is employed at every iteration for function and gradient evaluations.

In both variants, the Hessian ratio (5.2) is $R = 0.1$. Algorithm 5.2 employs an initial sample $\mathcal{S}_0$ of size 1% of the total training set, and the parameter in (3.9) is $\theta = 0.5$. In summary, both variants implement a subsampled Hessian Newton-CG method; Algorithm 5.2-Static is the method proposed in [7], using a fixed CG limit of 10 iterations, whereas Algorithm 5.2 is a refinement that allows a dynamic increase in the sample $\mathcal{S}_k$, in addition to the dynamic CG control.

Figure 5.1 displays the results of the comparative experiments in which Algorithm 5.2-Static is run with two values for the (fixed) sample size $|\mathcal{S}_k|$, namely $|\mathcal{S}_k| = N$ and $|\mathcal{S}_k| = 0.05N$. The horizontal axis plots the number of accessed data points, which is a representative measure of total computing time on a single machine; it takes into account function, gradient, and Hessian-vector products (see [7]). Instead of plotting the objective function J, which is a negative log likelihood function, the vertical axis plots the corresponding probability of correct classification, defined as $\exp(-J(w))$, so that the problem is presented in Figure 5.1 as a maximization problem. The gains obtained by using a dynamic sample size strategy are apparent in this experiment.

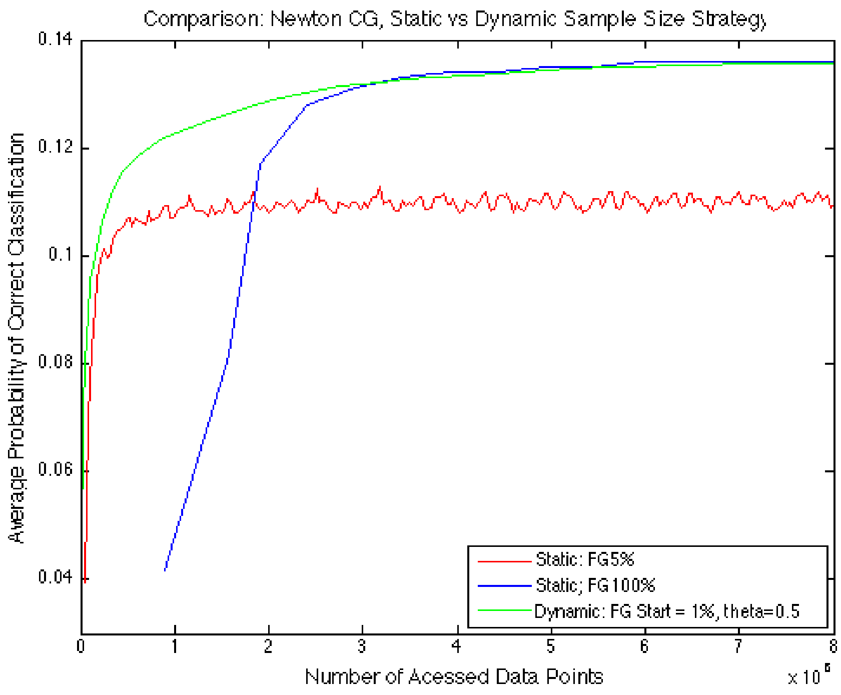

Figure 5.1: Comparison of Algorithm 5.2 (Dynamic) and Algorithm 5.2-Static, the latter with $|\mathcal{S}_k|$ equal to 100% and 5% of the training set size N

Next, we present data to measure the accuracy of the sample variances used in Algorithm 5.2, for this test problem. In Table 5.1, we report the values of the following quantities:

$$\begin{aligned}
A &: \quad \frac{\|\mathrm{Var}_{i\in\mathcal{S}}(\nabla \ell(w;i))\|_1}{|\mathcal{S}|} \\
B &: \quad \|\nabla J_{\mathcal{S}_k}(w) - \nabla J(w)\|_2^2 \\
Y &: \quad \frac{\|\mathrm{Var}_{i\in\mathcal{H}_k}\left(\nabla^2 \ell(w_k;i)\nabla J_{\mathcal{S}_k}(w_k)\right)\|_1}{|\mathcal{H}_k|\|\nabla J_{\mathcal{S}_k}(w_k)\|_2^2} \\
Z &: \quad \frac{\|[\nabla^2 J_{\mathcal{S}_k}(w_k) - \nabla^2 J_{\mathcal{H}_k}(w_k)]\nabla J_{\mathcal{S}_k}(w_k)\|_2^2}{\|\nabla J_{\mathcal{S}_k}(w_k)\|_2^2}.
\end{aligned}$$

In the light of the discussion in Section 3, we would like to observe whether the sample variance A is a reasonably good estimator for the error in the gradient, denoted by B. Similarly, based on the discussion in this section, we would like to compare quantities Y and Z.

Iter: k	A	B	Y	Z
1	0.041798576	0.039624485	0.015802753	0.016913783
2	0.040513236	0.03736721	0.033969807	0.034863612
3	0.036680293	0.035931856	0.027860283	0.019185297
4	0.028538358	0.028526197	0.017811839	0.017687138
5	0.02030112	0.01974306	0.015910543	0.014236308
⋮	⋮	⋮	⋮	⋮
13	0.001539071	0.001843071	0.002523053	0.002681272
14	0.000981444	0.001307763	0.0022572	0.002574446
15	0.000613751	0.000929579	0.000793887	0.001335829
16	0.000190385	0.00052025	0.000516926	0.00049049
17	0.000048608	0.000381851	0.00059497	0.0005979

Table 5.1: Analysis of Error Estimations

The table indicates that, for this test problem, the sample variances computed in the algorithm are reasonably accurate measures of the errors they estimate. We report only 17 iterations in Table 5.1 because at that point the dynamic sample strategy of Algorithm 5.2 has set $|\mathcal{S}_k| = N$. It is worth noting that, at that iteration, the objective function value $\exp(-J(w_{17})) \approx 0.123$, is close to its optimal value of 0.136. Therefore, as can be seen from Figure 5.1, most of the improvement in the objective function occurred while Algorithm 5.2 was operating in a dynamic sample regime.

In Figure 5.2, we report the effect of the parameter θ (see (3.9)) on the performance of Algorithm 5.2. In this experiment we set $R = 0.1$, and tested four values of θ. Note that as θ increases, the algorithm becomes more tolerant to larger variances in the batch gradient $\nabla J_{\mathcal{S}}$, and will tend to keep a sample size longer. We observe from Figure 5.2 that larger values of θ lead to an overall improvement in performance since they allow smaller sample sizes at the early iterations of the algorithm. The plots for $\theta = 0.5$ and

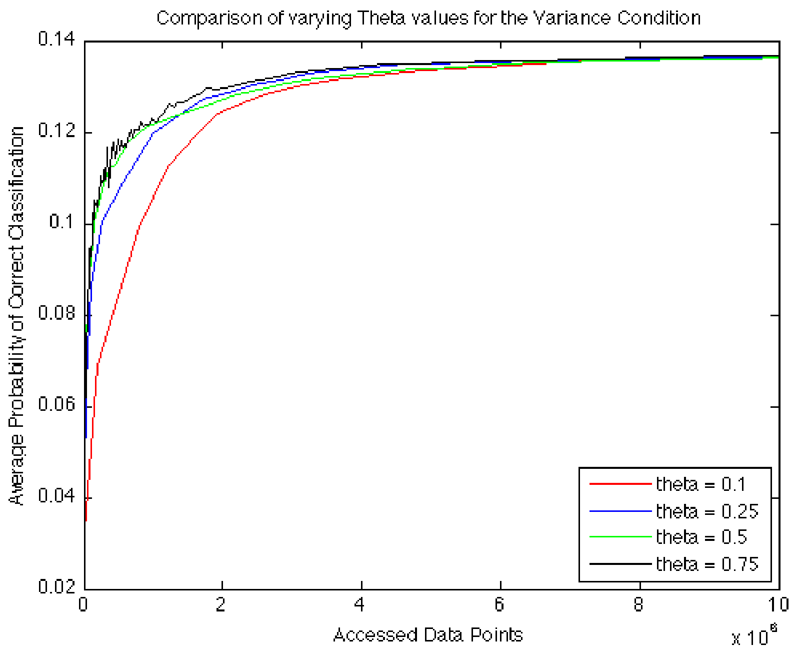

Figure 5.2: Comparison of varying θ values for Algorithm 5.2

0.75 are almost identical, but the behavior becomes more erratic as θ increases, as observed in the jagged line for $\theta = 0.75$. This is to be expected; although the increased flexibility allowed for accelerated performance in the early iterations, tolerating larger deviations in the approximate gradient is more likely to result in steps that do not yield progress in the target objective $J(w)$.

6 A Newton-CG Method for L$_1$ Regularized Models

Statistical learning models involving a very large number of parameters often benefit from the inclusion of an L$_1$ regularization term that forces some of these parameters to be zero. In this section, we describe a subsampled Hessian Newton-CG method for problems of this kind, in which the objective function (2.2) is replaced by

$$F(w) = \frac{1}{N}\sum_{i=1}^{N} l(f(w;x_i),y_i) + \nu\|w\|_1 = J(w) + \nu\|w\|_1, \tag{6.1}$$

where $\nu > 0$ is a (fixed) penalty parameter. Our mini-batch method chooses a subset $\mathcal{S} \subseteq \{1., ..., N\}$ of the training set, and applies an optimization algorithm to solve the related problem

$$\min_{w\in\mathbb{R}^m} F_{\mathcal{S}}(w) = J_{\mathcal{S}}(w) + \nu\|w\|_1, \tag{6.2}$$

where $J_{\mathcal{S}}$ is defined in (3.1).

A variety of first-order methods have been proposed for problem (6.1) (see for example [32, 15, 12, 4, 33, 14, 13]) and have proven to be very effective for many online applications. In keeping with the main theme of this paper, however, we consider only methods that incorporate second order information about the objective function and operate in a mini-batch setting. For simplicity, we assume in this section that the sample size $|\mathcal{S}|$ is fixed, so that we can focus on the challenges that arise in the design of a Hessian-free Newton method for the nonsmooth problem (6.1). (An extension of our method to the case when the sample size $|\mathcal{S}|$ changes at every iteration is the subject of a future paper.)

The proposed algorithm consists of two phases: an active-set prediction phase and a subspace minimization phase. The prediction phase estimates the variables that are zero at the solution, and identifies an *active orthant* where the subspace minimization is to take place. (Note that the objective (6.1) is smooth when restricted to any orthant in $\mathbb{R}^m$.) The subspace phase then minimizes a quadratic model of the objective function over the variables that are not active, to determine a direction along which progress in the objective can be made. A projected backtracking line search is performed to ensure that the new iterate remains in the same orthant as the current iterate, and yields a reduction in the objective $F_{\mathcal{S}}$. The subspace phase plays the dual role of accelerating convergence toward the solution while promoting the fast generation of sparse solutions.

There are several ways of implementing the two phases of the algorithm. One could employ iterative shrinkage [12] or the gradient projection method [24, 5] for the prediction phase; in this paper we chose a special form of the latter. For the subspace phase, one could minimize $F_{\mathcal{S}}$ [17, 9] instead of a quadratic model of this function [8, 20], but we choose to work with a model due to the high cost of evaluating the objective function.

6.1 Derivation of the Algorithm

The active-set prediction phase consists of a gradient projection search based on the subdifferential of the nonsmooth function $F(w)$; more specifically on the steepest descent direction at the current iterate w. We recall that, at a given a point w, the steepest descent direction for the non-smooth convex function F is defined as the solution of the problem

$$\min_{\|d\|_2 \le 1} \frac{dF(w+\alpha d)}{d\alpha}\Big|_{\alpha=0}.$$

For the function $F_{\mathcal{S}}$ defined in (6.2), this steepest descent direction, which we denote by $-\widetilde{\nabla} F_{\mathcal{S}}$, is given by

$$[\widetilde{\nabla} F_{\mathcal{S}}(w)]^i = \begin{cases} \frac{\partial J_{\mathcal{S}}(w)}{\partial w^i} + \nu & \text{if } w^i > 0, \\ \frac{\partial J_{\mathcal{S}}(w)}{\partial w^i} - \nu & \text{if } w^i < 0, \\ \frac{\partial J_{\mathcal{S}}(w)}{\partial w^i} + \nu & \text{if } w^i = 0, \text{ and } \frac{\partial J_{\mathcal{S}}(w)}{\partial w^i} < -\nu, \\ \frac{\partial J_{\mathcal{S}}(w)}{\partial w^i} - \nu & \text{if } w^i = 0, \text{ and } \frac{\partial J_{\mathcal{S}}(w)}{\partial w^i} > \nu, \\ 0 & \text{otherwise,} \end{cases} \tag{6.3}$$

where the superscript i indicates a component of a vector.

To identify the working orthant at the k-th iteration, we perform an "infinitesimal search" along the steepest descent direction, starting at the current iterate w_k. Let us define

$$\hat{w}_k(\alpha) \overset{\text{def}}{=} w_k - \alpha \widetilde{\nabla} F_{\mathcal{S}}(w_k), \quad \text{for } \alpha > 0. \tag{6.4}$$

The following cases arise: (i) If $w_k^i > 0$, then $\hat{w}_k^i(\alpha)$ will remain positive for sufficiently small α in (6.4), and in this case the active orthant is characterized by a positive component along the ith variable. If, on the other hand, $w_k^i < 0$, the active orthant corresponds to a negative component along the ith variable; (ii) When $w_k^i = 0$ and $\partial J_{\mathcal{S}}(w)/\partial w^i < -\nu$, we have from (6.4) that $\hat{w}_k^i(\alpha)$ is positive for small α. Alternatively, when $w_k^i = 0$ and $\partial J_{\mathcal{S}}(w)/\partial w^i > \nu$, then $\hat{w}_k(\alpha)$ becomes negative for small α; (iii) Finally, when $w_k^i = 0$ and $-\nu < \partial J_{\mathcal{S}}(w)/\partial w^i < \nu$, then $\hat{w}_k^i(\alpha) = 0$, indicating that this variable should be kept at zero.

Therefore, the orthant identified by an infinitesimal line search along the steepest descent direction is characterized by the vector

$$z_k^i = \begin{cases} 1 & w_k^i > 0, \\ -1 & w_k^i < 0, \\ 1 & w_k^i = 0, \text{ and } \frac{\partial J_{\mathcal{S}_k}(w_k)}{\partial w^i} < -\nu, \\ -1 & w_k^i = 0, \text{ and } \frac{\partial J_{\mathcal{S}_k}(w_k)}{\partial w^i} > \nu, \\ 0 & \text{otherwise.} \end{cases} \tag{6.5}$$

The significance of the vector z_k is evidenced by the fact that it defines the orthant

$$\Omega_k = \{d : \text{sign}(d^i) = \text{sign}(z_k^i)\}, \tag{6.6}$$

and in the relative interior of Ω_k, the function $\|w\|_1$ is differentiable, with its gradient given by z_k. Those variables with $z_k^i = 0$ constitute the active set $\mathcal{A}_k$, and will be kept at zero during the subspace phase, while the rest of the variables are free. Thus,

$$\mathcal{A}_k \triangleq \{\ i |\ z_k^i = 0\ \}. \tag{6.7}$$

Having identified the working orthant, the subspace minimization phase computes a solution of the following convex quadratic problem

$$\min_{d \in \mathbb{R}^m} \ m_k(d) \overset{\text{def}}{=} F_{\mathcal{S}_k}(w_k) + \widetilde{\nabla} F_{\mathcal{S}_k}(w_k)^T d + \tfrac{1}{2} d^T \nabla^2 J_{\mathcal{H}_k}(w_k) d \tag{6.8}$$

$$\text{s.t.} \ \ d^i = 0, \ i \in \mathcal{A}_k. \tag{6.9}$$

As in the previous section, the sample $\mathcal{H}_k$ used for the Hessian is chosen to be much smaller than the sample $\mathcal{S}_k$ used for the function and gradient. The minimization of the model $m_k(d)$ is performed by means the Hessian-free Newton-CG method, i.e., by applying the CG method to the linear system

$$\left[Y_k^T \nabla^2 J_{\mathcal{H}_k}(w_k) Y_k\right] d^Y = -Y_k^T \widetilde{\nabla} F_{\mathcal{S}_k}(w_k), \tag{6.10}$$

where Y_k is a basis spanning the space of free variables, and d^Y is a vector of length $n-|\mathcal{A}_k|$. Specifically, we define Y_k to be the $n \times (n - |\mathcal{A}_k|)$ matrix where each column has a 1 in the position of a given free variable and a 0 elsewhere. Given the approximate solution of (6.10) d^Y, we define the search direction of the algorithm to be $d_k = Y_k d^Y$.

Next, we perform a projected backtracking line search, as follows. Let $P(\cdot)$ denote the orthogonal projection onto the orthant Ω_k. Thus,

$$P(w^i) = \begin{cases} w^i & \text{if } \operatorname{sign}(w^i) = \operatorname{sign}(z_k^i) \\ 0 & \text{otherwise.} \end{cases} \tag{6.11}$$

The line search computes a steplength α_k to be the largest member of the sequence $1, 1/2, 1/4, ...$ such that

$$F_{\mathcal{S}_k}(P[w_k + \alpha_k d_k]) \le F_{\mathcal{S}_k}(w_k) + \sigma \widetilde{\nabla} F_{\mathcal{S}_k}(w_k)^T (P[w_k + \alpha_k d_k] - w_k), \tag{6.12}$$

where $\sigma \in (0,1)$ is a given constant. The new iterate is defined as $w_{k+1} = P[w_k + \alpha_k d_k]$. This method is summarized in Algorithm 6.1

While the orthant identified at every iteration of Algorithm 6.1 coincides with that generated by the OWL method [2], the two algorithms differ in two respects. In the subspace minimization phase, OWL computes a limited memory BFGS step in the space of free variables, while Algorithm 6.1 employs the subsampled Hessian Newton-CG method. The second difference between the two algorithms, is that the OWL algorithm performs an *alignment* of the subspace step d_k with the steepest descent direction (6.3) by resetting

$$[d_k]^i \leftarrow \begin{cases} [d_k]^i & \text{if } \operatorname{sign}([d_k]^i) = \operatorname{sign}([-\widetilde{\nabla} F_{\mathcal{S}_k}(w_k)]^i), \\ 0 & \text{otherwise} \end{cases} \tag{6.14}$$

at every iteration. The alignment is motivated by the need to achieve global convergence properties, but our computational experience indicates that it slows down the iteration. Moreover, the convergence proof given in [2] is not correct, due to an invalid assumption made in Proposition 4. A simple counter-example can be constructed that contradicts this assumption, based on the non-expansive property of projections and the realignment step. Since there are more appropriate mechanisms for guaranteeing convergence, as we explain at the end of section 6.2, we have omitted the alignment procedure (6.14) in our algorithm.

6.2 Numerical Tests

To assess the effectiveness of Algorithm 6.1, we test it on a larger version of the speech recognition problem described in section 5.2. The new problem has $N = 191607$ training points and $m = 30315$ parameters in w. This problem was chosen to admit a useful sparse solution: while the training set is only 14% larger than that used in section 5.2, there are almost 3 times more parameters. We compare the following two methods in Figures 6.1 and 6.2:

Algorithm 6.1: Newton-CG Algorithm for Solving Problem (6.1)

```
Initialize: Choose an initial iterate $w_0$, constants $\eta, \sigma \in (0,1)$, a CG iteration limit $\max_{cg}$, the regularization constant $\nu > 0$ and initial samples $\mathcal{H}_0$ and $\mathcal{S}_0$ such that $|\mathcal{H}_0| < |\mathcal{S}_0|$.
for iteration $k = 0, \cdots$, and until a convergence test is satisfied:
    1. Determine the active set $\mathcal{A}_k$ and the active orthant characterized by $z_k$, as given by (6.7) and (6.5), respectively.
    2. Subspace Minimization Phase.
        2.1 Compute $\widetilde{\nabla} F_{\mathcal{S}_k}(w_k) = \nabla J_{\mathcal{S}_k}(w_k) + \nu z_k$, where $z_k$ is given in (6.5).
        2.2 Apply the conjugate gradient method to compute an approximate solution $d_k$ of the linear system (6.10). The CG iteration is terminated when either $\max_{cg}$ iterations have been performed or when the residual $r_k$ in (6.10) satisfies
                $r_k \leq \eta \| Y_k^T \widetilde{\nabla} F_{\mathcal{S}_k}(x_k) \|$.    (6.13)
    3. Perform line search. Determine the largest value $\alpha_k \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ such that condition (6.12) is satisfied. Set
                $w_{k+1} = P[w_k + \alpha_k d_k]$
       where $P$ is defined in (6.11)
    4. Resample the data sets $\mathcal{H}_{k+1}$, $\mathcal{S}_{k+1}$ so that $|\mathcal{H}_{k+1}| = |\mathcal{H}_0|$ and $|\mathcal{S}_{k+1}| = |\mathcal{S}_0|$.
```

i) The OWL algorithm using two values of the limited memory parameter, namely 5 and 20. We tested the implementation provided in LibLBFGS

   `http://www.chokkan.org/software/liblbfgs/`

ii) The subsampled Hessian Newton L_1 method of Algorithm 6.1 with $\max_{cg} = 10$, $R = |\mathcal{H}_k|/|\mathcal{S}_k| = 0.05$ and $|\mathcal{S}_k| = N$.

For both methods, the initial point is $w_0 = 0$. Figure 6.1 plots the objective function F, and Figure 6.2 plots the number of non-zeros in the solution w_k, both as a function CPU time (more precisely, the number of training points accessed). We observe that although Algorithm 6.1 is only moderately faster than OWL in terms of improvement in the objective function, it is much more effective at generating a sparse solution quickly.

We have mentioned that Algorithm 6.1 and the OWL method employ the same orthant identification mechanism at each iteration, but our derivation views the use of the steepest descent direction to select the orthant as a version of a gradient projection step. This has the advantage of suggesting variants of the algorithm and mechanisms for establishing global

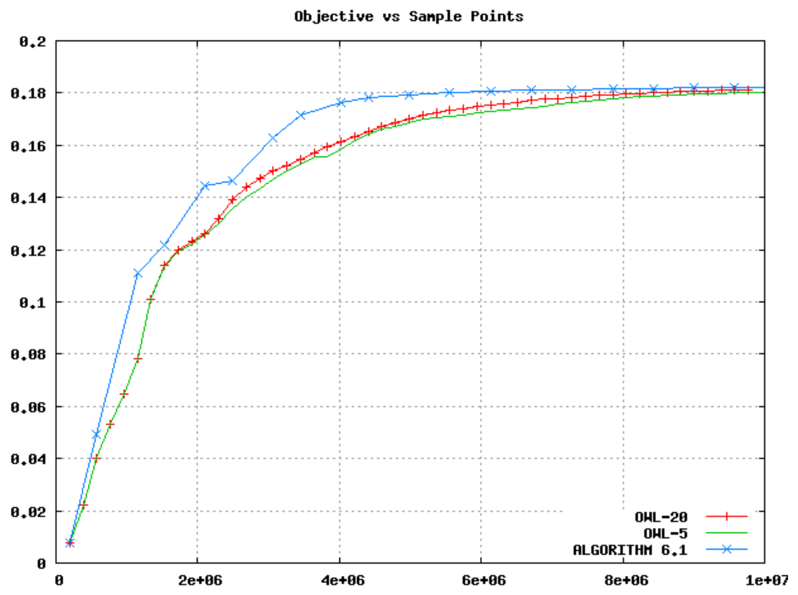

Figure 6.1: Comparison of OWL (with memory 5 and 20) and Algorithm 6.1. The vertical axis plots the objective function, and the horizontal axis, the number of accessed data points

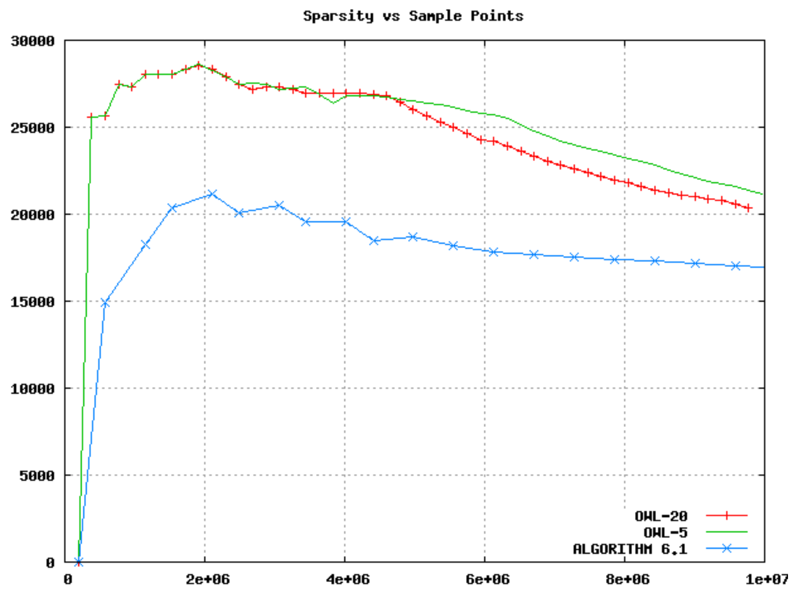

Figure 6.2: Comparison of OWL and Algorithm 6.1. The vertical axis plots the number of non-zeros in the solution vector, and the horizontal axis, the number of accessed data points.

convergence. For example, if the "infinitesimal" line search was replaced by a projected search in which a quadratic model was minimized, one might obtain a better prediction of the active orthant, particularly in the early stages, and one could establish convergence. The design of a more elaborate gradient projection search is, however, not simple, and we defer an investigation of this topic, as well as the design of a procedure for varying the sample size $|\mathcal{S}_k|$, to a future paper.

7 Related Work

Variable sample size methods for stochastic optimization have been a subject of previous research. Due to the dynamic nature of varying sample sizes, the traditional convergence results developed for sample average approximation require additional conditions. Shapiro and Wardi [29] present convergence results in the framework of point-to-set mappings, whereas further examination by Kleywegt et al. [19] and Shapiro and Homem-de-Mello [28] provide results that show convergence for specific classes of problems.

The premise of our dynamic sample size algorithm is to efficiently generate directions that decrease the target objective function, sufficiently often. In [27], Shapiro and Homem-de-Mello follow a similar strategy, presenting a variable sample size methodology where the determination of the new increased sample size is computed using a confidence ellipsoid for the stochastic gradient estimate, denoted in this paper as $\nabla J_{\mathcal{S}_k}(w_k)$. However, for large scale problems, this methodology is not computationally feasible given that the sample covariance matrix is required for the determination of the confidence ellipsoid. Further, if the sample size is not large enough, the corresponding covariance matrix is poorly estimated and could result in very large values for the suggested sample size that must be safeguarded.

Alternative strategies have been proposed that are fairly diverse in the implementation of variable sample size strategies, all of which have proven convergence properties. In [18], Homem-de-Mello presents a variable sample size method that utilizes only function values, within the framework of a pure random search method. The iteration progresses, provided the samples satisfy corresponding statistical t-tests, to ensure that the algorithm can determine a good sample size. Deng and Ferris [11] construct a variable sample size methodology for simulation based optimization, where no analytical form for the function exists. Along the lines of utilizing simulation, Bastin et al. [3] also suggest a varying sample size methodology that uses Monte Carlo methods. The main strategy of this algorithm is to adapt the traditional deterministic trust-region approach to handle both stochasticity and adaptive sample sizes. The strategy for increasing the sample is quite different from the one presented in this paper; it relies on function values and a system of sample size updates analogous to update rules for trust region sizes.

8 Final Remarks

In conclusion, we have proposed advancements to the topic of subsampling as presented in [7]. We firstly presented a dynamic sample size gradient based strategy for solving large scale machine learning problems, using a batch optimization framework. There are two components of this sampling strategy: the first being the condition as to when the sample size should increase, and the second, being the rule for choosing the new sample size. The strategic development of this sampling strategy focused on utilizing sample variance estimates to ensure that the algorithm makes sufficient progress on the target objective function. We presented convergence results for the deterministic case, as well as competitive complexity results for the stochastic case. Furthermore, we have presented numerical results using a speech recognition problem, illustrating the improved performance of our dynamic sample size strategy, in comparison to the previous sub-sampling algorithm given in [7].

The second contribution of this paper is the extension of the Hessian subsampling strategy to the L1-regularized problem. We present this strategy as a two phase method, composed of an active set identification phase using an infinitesimal line search, and a subspace minimization phase that utilizes the Hessian subsampling technique. We present numerical results for a sparse version of the speech recognition problem, and we have shown that our algorithm is able to outperform the OWL algorithm presented in [2], both in terms of sparsity and objective value.

Some questions still remain in utilizing sampling strategies for the L1 regularized problem. The use of a different active set identification mechanism, other than an infinitesimal line search, could yield a more useful subspace for performing the second minimization step. Secondly, the design of dynamic sampling techniques for evaluating the function and the gradient is a challenging endeavor in the L1 context, and remains to be explored. Further research into these questions could lead to the design of more powerful methods for solving large scale regularized problems.

Acknowledgments. The authors thank Will Neveitt for many valuable insights and suggestions, and Tito Homem-de-Mello for several useful comments on this work.

References

[1] A. Agarwal and J.C. Duchi. Distributed delayed stochastic optimization. *Arxiv preprint arXiv:1104.5525*, 2011.

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

[3] Bastin, F. , C. Cirillo, P.L. Toint. An adaptive monte carlo algorithm for computing mixed logit estimators. *Computational Management Sciences*, 3(1):55–79, 2006.

[4] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[5] D. P. Bertsekas. On the Goldstein-Levitin-Poljak gradient projection method. *IEEE Transactions on Automatic Control*, AC-21:174–184, 1976.

[6] Leon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 161–168. MIT Press, Cambridge, MA, 2008.

[7] 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.

[8] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572, 1991.

[9] Y.H. Dai and R. Fletcher. Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. *Numerische Mathematik*, 100(1):21–47, 2005.

[10] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *Arxiv preprint arXiv:1012.1367*, 2010.

[11] Deng, G. and M. C. Ferris. Variable-number sample-path optimization. *Mathematical Programming*, 117(1-2):81–109, 2009.

[12] D.L. Donoho. De-noising by soft-thresholding. *Information Theory, IEEE Transactions on*, 41(3):613–627, 1995.

[13] J. Duchi, S. Shalev-Shwartz, Y. Singer, and A. Tewari. Composite objective mirror descent. In *Proceedings of the Twenty Third Annual Conference on Computational Learning Theory*. Citeseer, 2010.

[14] J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *The Journal of Machine Learning Research*, 10:2899–2934, 2009.

[15] M.A.T. Figueiredo, R.D. Nowak, and S.J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):586–597, 2007.

[16] John E. Freund. *Mathematical Statistics*. Prentice Hall, 1962.

[17] W. W. Hager and H. Zhang. A new active set algorithm for box constrained optimization. *SIOPT*, 17(2):526–557, 2007.

[18] Homem-de-Mello, T. Variable-sample methods for stochastic optimization. *ACM Transactions on Modeling and Computer Simulation*, 13(2):108–133, 2003.

[19] Kleywegt, A.J., A. Shapiro and T. Homem-de-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2001.

[20] C.J. Lin, J.J. Moré, et al. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, 9(4):1100–1127, 1999.

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

[22] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Math. Program.*, 120(1):221–259, 2009.

[23] F. Niu, B. Recht, C. Ré, and S.J. Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. 2011.

[24] B. T. Polyak. The conjugate gradient method in extremal problems. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 9:94–112, 1969.

[25] B.T. Polyak and A.B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30:838, 1992.

[26] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[27] Shapiro, A. and T. Homem-de-Mello. A simulation-based approach to two-stage stochastic programming with recourse. *Mathematical Programming*, 81:301–325, 1998.

[28] Shapiro, A. and T. Homem-de-Mello. On the rate of convergence of optimal solutions of monte carlo approximations of stochastic programs. *SIAM Journal on Optimization*, 11(1):70–86, 2000.

[29] Shapiro, A. and Y. Wardi. Convergence of stochastic algorithms. *Mathematics of Operations Research*, 21(3):615–628, 1996.

[30] SVN Vishwanathan, N.N. Schraudolph, M.W. Schmidt, and K.P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the 23rd international conference on Machine learning*, pages 969–976. ACM, 2006.

[31] S.J. Wright. Accelerated block-coordinate relaxation for regularized optimization. Technical report, Computer Science Department, University of Wisconsin, 2010.

[32] S.J. Wright, R.D. Nowak, and M.A.T. Figueiredo. Sparse reconstruction by separable approximation. *Signal Processing, IEEE Transactions on*, 57(7):2479–2493, 2009.

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