HYBRID DETERMINISTIC-STOCHASTIC METHODS FOR DATA FITTING

MICHAEL P. FRIEDLANDER* AND MARK SCHMIDT†

Abstract. Many structured data-fitting applications require the solution of an optimization problem involving a sum over a potentially large number of measurements. Incremental gradient algorithms (both deterministic and randomized) offer inexpensive iterations by sampling only subsets of the terms in the sum. These methods can make great progress initially, but often slow as they approach a solution. In contrast, full gradient methods achieve steady convergence at the expense of evaluating the full objective and gradient on each iteration. We explore hybrid methods that exhibit the benefits of both approaches. Rate of convergence analysis and numerical experiments illustrate the potential for the approach.

1. Introduction.

Data fitting applications are often typified by optimization problems of the form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x) := \frac{1}{M}\sum_{i=1}^{M} f_i(x), \tag{1.1}$$

where each function f_i corresponds to a single observation or measurement, and models the misfit for a given choice of parameters x. The aim is to choose parameters to minimize the misfit across all measurements. The canonical example is least-squares, and in that case,

$$f_i(x) = \tfrac{1}{2}(a_i^T x - b_i)^2.$$

This misfit model corresponds to a linear model with Gaussian errors on the measurements b. For applications where the measurements b are binary, a more appropriate model is logistic regression, described by the choice

$$f_i(x) = \log(1 + \exp[-b_i a_i^T x]).$$

These are both simple cases of the more general maximum-likelihood problem. The maximum-likelihood approach gives rise to separable problems like (1.1) whenever the measurements (a_i, b_i) are assumed to be independent and identically distributed.

If the number of measurements M is very large, then evaluating $f(x)$ and $\nabla f(x)$ can be expensive, even if the individual f_i are simple functions. However, there is often a large amount of uniformity in the measurements, which means that a full evaluation of $f(x)$ and $\nabla f(x)$ may be unnecessary to make progress in solving (1.1). This motivates *incremental* gradient methods, where each iteration only evaluates the gradient with respect to a single f_i [3, §3.2].

Incremental gradient methods enjoy an iteration cost that is M times faster than full gradient methods because the iterations are independent of M. Thus, in the time it

*Department of Computer Science, University of British Columbia, Vancouver V6T 1Z4, B.C., Canada (mpf@cs.ubc.ca). The work of this author was partially supported by the DNOISE II CRD grant from the Natural Sciences and Engineering Research Council (NSERC).

†INRIA - SIERRA team, Laboratoire d'Informatique de l'Ecole Normale Superieure (INRIA/ENS/CNRS UMR 8548), 23 avenue d'Italie, 75214 Paris CEDEX 13, France (mark.schmidt@inria.fr). The work of this author was partially supported by the CRD DNOISE II NSERC grant, and the SIERRA grant from the European Research Research Council (SIERRA-ERC-23999).

April 12, 2011

takes to make one full gradient iteration, the incremental gradient method can achieve M iterations, which often results in rapid initial progress. However, the number of iterations to reach the same level of accuracy may be much higher. Indeed, because of their faster convergence rate, full gradient methods must eventually dominate incremental gradient methods.

Our aim is to develop a method that exhibits the benefits of these two extremes. The approach is based on starting with iterations that resemble an incremental gradient approach and use relatively few measurements to approximate the gradient; as the iterations proceed, the algorithm gradually increases the number of measurements. This preserves the rapid initial progress of incremental gradient methods without sacrificing the convergence rate of full gradient methods.

1.1. Assumptions and notation. We make the blanket assumption throughout that a minimizer x_* of f always exists, that the functions $f_i : \mathbb{R}^n \to \mathbb{R}$ are continuously differentiable, and that the overall gradient of f is uniformly Lipschitz continuous, i.e., for some positive L,

$$\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\| \quad \text{for all} \quad x, y \in \mathbb{R}^n. \tag{1.2a}$$

We also assume that f is strongly convex with (positive) parameter μ:

$$f(y) \ge f(x) + (y - x)^T \nabla f(x) + \tfrac{1}{2}\mu\|y - x\|^2 \quad \text{for all} \quad x, y \in \mathbb{R}^n. \tag{1.2b}$$

If f is twice-continuously differentiable, then these assumptions are equivalent to the condition that the eigenvalues of the Hessian are uniformly bounded above and below:

$$\mu I \preceq \nabla^2 f(x) \preceq LI.$$

The ratio $L/\mu \ge 1$ is known as the *condition* number of f [25, §2.13].

We describe two versions of linear convergence of function values. The first is denoted as *weak* linear convergence, and is characterized by a bounding sequence on the function values:

$$f(x_k) - f(x_*) = \mathcal{O}(\sigma^k) \quad \text{for some} \quad \sigma < 1 \tag{1.3}$$

for all k. This is a non-asymptotic version of R-linear convergence. (See, for example, [27, §A.2].) The second is denoted as *strong* linear convergence, and characterizes the decrease of the function value at every iteration k:

$$f(x_{k+1}) - f(x_*) \le \sigma[f(x_k) - f(x_*)] \quad \text{for some} \quad \sigma < 1. \tag{1.4}$$

We emphasize that this inequality applies to all iterations of the algorithm, and is a non-asymptotic version of Q-linear convergence. Note that (1.4) implies (1.3), though the converse is not implied.

All of the convergence results that we analyze are described only in terms of convergence of the objective function values, and not the iterates x_k. This is sufficient, however, because the strong convexity assumption allows us to directly deduce a convergence rate on x_k to x_* via the corresponding function values. In particular, strong convexity of f implies

$$\frac{\mu}{2}\|x_k - x_*\|^2 \le f(x_k) - f(x_*).$$

Thus, the rate at which the error $\|x_k - x_*\|^2$ converges is at least as fast as the rate at which $f(x_k)$ converges to the optimal value $f(x_*)$.

1.2. Gradient descent with error. In the most basic version of the algorithm that we consider, each iterate is computed via the update

$$x_{k+1} = x_k - \alpha_k g(x_k) \tag{1.5}$$

for some step size α_k; the search direction

$$g(x_k) := \nabla f(x_k) + e_k, \tag{1.6}$$

is an approximation of the gradient, and e_k is the residual in its computation. Evidently, the full gradient method (i.e., steepest descent) corresponds to the case where $e_k \equiv 0$, which means that the gradient is exactly computed at each iteration.

The stochastic approximation method deals with the case where the residual e_k is a random variable, see [3]. In this context, we typically assume that $\mathbb{E}[e_k] = 0$ and $\mathbb{E}[\|e_k\|^2] \leq B$ for some constant B, which are sufficient to guarantee that the iterates converge in a probabilistic sense (for a suitable choice of decreasing step sizes). Incremental gradient methods are a special case of stochastic approximation. Here, rather than computing the full gradient $\nabla f(x)$ on each iteration, a function f_i is randomly selected among $i \in \{1, \dots, M\}$, and the gradient estimate is constructed as $g(x_k) = f_i(x_k)$; a standard assumption is that

$$\|\nabla f_i(x)\|^2 \leq \beta_1 + \beta_2 \|\nabla f(x)\|^2 \quad \text{for all } x \text{ and } i = 1, \dots, M, \tag{1.7}$$

for some constants $\beta_1 \geq 0$ and $\beta_2 \geq 1$. This implies that $\|e_k\|$ is bounded as a function of the true gradient of the objective; see, e.g., [3, §4.2].

In this work, we consider scenarios where you can control the error by, on each iteration, specifying a bound B_k on the error. In particular, we characterize convergence under the gradient-with-error algorithm (1.5)–(1.6) under two different conditions:

$$\|e_k\|^2 \leq B_k \quad \text{or} \quad \mathbb{E}[\|e_k\|^2] \leq B_k. \tag{1.8}$$

Note that our analysis applies whether the noise is deterministic or stochastic, and we do not assume that the noise has zero mean.

This paper is divided into five distinct components.

Weak linear convergence with generic bounds (§2.1). We analyze the convergence rate under a generic sequence $\{B_k\}$. Our results imply that for any (sub)linearly decreasing sequence $\{B_k\}$, the algorithm has a weak (sub)linear convergence rate. In the expected-error version of (1.8), the convergence rate is described in terms of the expected function value.

Strong linear convergence with particular bounds (§2.2). We describe a particular construction of the sequence $\{B_k\}$ that ensures that the algorithm has a strong linear convergence rate. The rate achieved under this sequence can be arbitrarily close to the rate of the standard gradient method without error, without requiring an exact gradient calculation on any iteration.

Application to incremental gradient (§3). We use a growing-batch strategy as a mechanism for controlling the error in the estimated gradient and achieving a linear rate. In effect, choosing the batch size allows us to control the expected error, as in (1.8). By growing the batch size sufficiently fast, we implicitly set the rate at which $B_k \to 0$, and hence the overall rate of the algorithm.

A practical quasi-Newton implementation (§4). We show how the convergence rate is changed under a scaling of the gradient, as in quasi-Newton methods (§4.1). Subsequently, we describe a practical implementation of the ideas based on a limited-memory quasi-Newton approximation and a heuristic line search.

Numerical results (§5). We evaluate the implementation on a variety of data fitting applications (comparing to full gradient and incremental gradient methods).

1.3. Related work. Our approach is based on bridging the gap between two ends of a spectrum, where incremental gradient methods are at the end of "cheap iterations with slow convergence", and full gradient methods are "expensive iterations with fast convergence". In particular, incremental gradient methods achieve an expected sublinear convergence rate on the expected value of $f(x_k)$, i.e.,

$$\mathbb{E}[f(x_k) - f(x_*)] = \mathcal{O}(1/k),$$

where the iterations are described by (1.5) for $\alpha_k = \mathcal{O}(1/k)$ [24, §2.1]. In fact, among all first-order methods, this is the best possible dependency on k given only a first-order stochastic oracle; thus a linear rate is not possible [23, §14.1].

Consider the basic gradient descent iteration (1.5) with a fixed step size $\alpha_k = 1/L$ and $g(x_k) = \nabla f(x_k)$. It is well known that this algorithm has a strong linear convergence rate, and satisfies the per-iteration decrease in (1.4) with $\sigma = 1 - \mu/L$; see [16, §8.6].

Some authors have analyzed incremental gradient methods with a constant step size [22]. Although this strategy does not converge to the optimal solution, it does converge at a linear rate to a neighborhood of the solution (where the size of the neighborhood increases with the step size).

In the context of incremental gradient methods, several other hybrid methods have been proposed that achieve a linear convergence rate. The works of Bertsekas [2] and Blatt et al. [6] are the closest in spirit to our proposed approach. However, the convergence rates for these methods treat full passes through the data as iterations, similar to the full gradient method. Further, there are numerical difficulties in evaluating certain sequences associated with the method of [2], while the method of [6] may require an excessive amount of memory.

This is not the first work to examine a growing-batch strategy. This type of strategy appears to be a "folk" algorithm used by practitioners in several application domains, and it is explicitly mentioned in an informal context by some authors such as Bertsekas and Tsitsiklas [3, page 113]. This is the first work, that we are of, that presents a theoretical analysis of the technique and that proposes a practical large-scale quasi-Newton implementation along with an experimental evaluation.

Gradient descent with a decreasing sequence of errors in the gradient measurement was previously analyzed by Luo and Tseng [17], and they present analogous weak and strong linear convergence results depending on the sequence of bounds on the noise. Our analysis extends this previous work in several ways:

- The weak linear convergence rate shown in [17] requires a strict decrease in the objective function at every iteration In contrast, our analysis in §2.1 does not require this assumption, and allows for the (realistic) possibility that the noisy gradient can lead to an increase in the objective function on some iterations.
- The strong linear convergence rate shown in [17] only holds asymptotically, while the construction we give (see §2.2) leads to a non-asymptotic rate of the form (1.4) that applies to all iterations of the algorithm.
- Luo and Tseng [17] consider deterministic errors in the gradient measurement that can be bounded in an absolute sense; we also consider the more general scenario where the error is stochastic and can only be bounded in expectation.

1.4. Reproducible research. Following the discipline of reproducible research, the source code and data files required to reproduce the experimental results of this paper can be downloaded from

`http://www.cs.ubc.ca/labs/scl/FriedlanderSchmidt2011`.

2. Convergence analysis.

Our convergence analysis first considers a basic first-order method with the constant step size $\alpha_k = 1/L$. In §4.1 we consider variants with a scaled search direction, like quasi-Newton methods.

The following intermediate result establishes an upper bound on the objective value at each iteration in terms of the residual in the computed gradient.

LEMMA 2.1. *At each iteration k of algorithm* (1.5)*, with $\alpha_k \equiv 1/L$,*

$$f(x_{k+1}) - f(x_*) \le (1-\mu/L)[f(x_k) - f(x_*)] + \frac{1}{2L}\|e_k\|^2. \tag{2.1}$$

Proof. It follows from assumptions (1.2) that the following inequalities hold:

$$f(y) \le f(x) + (y-x)^T\nabla f(x) + \frac{L}{2}\|y-x\|^2, \tag{2.2a}$$

$$f(y) \ge f(x) + (y-x)^T\nabla f(x) + \frac{\mu}{2}\|y-x\|^2. \tag{2.2b}$$

Use $x = x_k$ and $y = x_k - (1/L)g(x_k)$ in (2.2a) and simplify to obtain

$$f(x_k - (1/L)g(x_k)) \le f(x_k) - \frac{1}{L}g(x_k)^T\nabla f(x_k) + \frac{1}{2L}\|g(x_k)\|^2.$$

Next, use the definitions of x_{k+1} and $g(x_k)$ (cf. (1.5)–(1.6)) in this expression to obtain

$$\begin{aligned} f(x_{k+1}) &\le f(x_k) - \frac{1}{L}(\nabla f(x_k) + e_k)^T\nabla f(x_k) + \frac{1}{2L}\|\nabla f(x_k) + e_k\|^2 \\ &= f(x_k) - \frac{1}{L}\|\nabla f(x_k)\|^2 - \frac{1}{L}\nabla f(x_k)^T e_k \\ &\qquad + \frac{1}{2L}\|\nabla f(x_k)\|^2 + \frac{1}{L}\nabla f(x_k)^T e_k + \frac{1}{2L}\|e_k\|^2 \\ &= f(x_k) - \frac{1}{2L}\|\nabla f(x_k)\|^2 + \frac{1}{2L}\|e_k\|^2. \end{aligned} \tag{2.3}$$

We now use (2.2b) to derive a lower bound on the norm of $\nabla f(x_k)$ in terms of the optimality of $f(x_k)$. Do this by minimizing both sides of (2.2b) with respect to y: by definition, the minimum of the left-hand side is achieved by $y = x_*$; the minimizer of the right-hand side is given by $y = x - (1/\mu)\nabla f(x)$. Thus, we have for any x that

$$\begin{aligned} f(x_*) &\ge f(x) - \frac{1}{\mu}\nabla f(x)^T\nabla f(x) + \frac{1}{2\mu}\nabla f(x)^T\nabla f(x) \\ &= f(x) - \frac{1}{2\mu}\|\nabla f(x)\|^2. \end{aligned}$$

Re-arranging and specializing to the case where $x = x_k$,

$$\|\nabla f(x_k)\|^2 \ge 2\mu[f(x_k) - f(x_*)]. \tag{2.4}$$

Subtract $f(x_*)$ from both sides of (2.3) and use (2.4) to get

$$\begin{aligned} f(x_{k+1}) - f(x_*) &\le f(x_k) - f(x_*) - \frac{\mu}{L}[f(x_k) - f(x_*)] + \frac{1}{2L}\|e_k\|^2 \\ &= (1-\mu/L)[f(x_k) - f(x_*)] + \frac{1}{2L}\|e_k\|^2, \end{aligned}$$

which gives the required result. □

As an aside, note that (2.3) shows that the objective decreases monotonically, i.e., $f(x_{k+1}) < f(x_k)$, if $\|e_k\| < \|\nabla f(x_k)\|$. However, in general we do not require this condition.

2.1. Weak linear convergence. In this section we show that if $\{B_k\}$ is any (sub)linearly convergent sequence, then algorithm (1.5) has a (sub)linear convergence rate. This result reflects that the convergence rate of the approximate gradient algorithm is not better than the rate at which the noise goes to zero, and of course is also not better than the rate of the noiseless algorithm.

Theorem 2.2 (Weak convergence rate under absolute error bounds). *Suppose that* $\|e_k\|^2 \le B_k$, *where*

$$\lim_{k\to\infty} B_{k+1}/B_k \le 1. \tag{2.5}$$

Then at each iteration of algorithm (1.5) *with* $\alpha_k \equiv 1/L$,

$$f(x_k) - f(x_*) = \mathcal{O}(C_k), \tag{2.6}$$

where $C_k = \max\{B_k, (1-\mu/L+\epsilon)^k\}$ *for any* $\epsilon > 0$ *such that* $\epsilon < \mu/L$.

Proof. Let $\rho = 1-\mu/L$. Because $\|e_k\|^2 \le B_k$, Lemma 2.1 implies

$$f(x_{k+1}) - f(x_*) \le \rho[f(x_k) - f(x_*)] + \frac{1}{2L}B_k.$$

Applying this recursively,

$$f(x_k) - f(x_*) = \mathcal{O}(\rho^k) + \mu_k, \tag{2.7}$$

where

$$\mu_k := \sum_{i=0}^{k-1} \mathcal{O}(\rho^{k-i-1}B_i).$$

Observe that $\mu_{k+1} = B_k + \rho\mu_k$. Let $\gamma := \lim_{k\to\infty} B_{k+1}/B_k$.

Consider the first case where $\gamma > \rho$. Because the term $\mathcal{O}(\rho^k)$ in (2.7) is in $\mathcal{O}(C_k)$, we only need to show that $\mu_k = \mathcal{O}(C_k)$. We do this by induction. Because $\gamma > \rho$, it follows from (2.5) and the definition of C_k that there exists some k such that $C_{k'+1}/C_{k'} > \rho$ for all $k' \ge k$. Let k be the first iteration with this property, and choose ξ such that

$$\mu_{k'} \le \xi C_{k'} \text{ for all } k' \le k, \quad \text{and} \quad \xi(C_{k+1}/C_k - \rho) \ge 1.$$

The former is always possible because the $\mu_{k'}$ are finite. We then have that

$$\mu_{k+1} = C_k + \rho\mu_k \le C_k + \rho\xi C_k \le \xi(C_{k+1}/C_k - \rho)B_k + \rho\xi C_k = \xi C_{k+1}.$$

Thus, $\mu_k = \mathcal{O}(C_k)$ for all k, as required.

Next consider the case where $\gamma \le \rho$. Because the term $\mathcal{O}(\rho^k)$ in (2.7) is linearly convergent with rate ρ, we only need to show that $\mu_k \to 0$ linearly with rate ρ, i.e., $\mu_{k+1}/\mu_k \to \rho$ as $k \to \infty$. Note that $\mu_{k+1}/\mu_k = B_k/\mu_k + \rho$, and thus it is sufficient to show that $B_k/\mu_k \to 0$, i.e., $\mu_k/B_k \to \infty$. Expanding μ_k, and using the fact that $B_{k-1}/B_k = \gamma$, $B_{k-2}/B_k = \gamma^2$, etc., we obtain

$$\begin{aligned}\lim_{k\to\infty}\frac{\mu_k}{B_k} &= \lim_{k\to\infty}\rho^0\frac{B_{k-1}}{B_k} + \rho^1\frac{B_{k-2}}{B_k} + \cdots + \rho^{k-1}\frac{B_0}{B_k}\\ &= (1/\rho)\lim_{k\to\infty}\left[\rho^1\frac{B_{k-1}}{B_k} + \rho^2\frac{B_{k-2}}{B_k} + \cdots + \rho^k\frac{B_0}{B_k}\right]\\ &= (1/\rho)\sum_{i=1}^{\infty}(\rho/\gamma)^i.\end{aligned}$$

This sequence diverges because $\gamma \le \rho$. Thus, the right-hand side of (2.7) converges to 0 Q-linearly with rate $\rho = 1-\mu/L$. The required result then follows from the observation that every sequence that converges to 0 Q-linearly with rate ρ is in $\mathcal{O}([\rho+\epsilon]^k)$ for any $\epsilon > 0$, where $\rho+\epsilon < 1$, i.e., $\epsilon < \mu/L$. □

An implication of this result is that the algorithm has a linear convergence rate if B_k converges linearly to zero. For example, if $B_k = \mathcal{O}(\gamma^k)$ with $\gamma < 1$, then

$$f(x_k) - f(x_*) = \mathcal{O}(\sigma^k),$$

where $\sigma = \max\{\gamma, (1-\mu/L+\epsilon)\})$ for any positive $\epsilon < \mu/L$.

Theorem 2.2 also yields a convergence rate in scenarios where the high cost of computing an accurate gradient might make it appealing to allow the error in the gradient measurement to decrease sublinearly. For example, if $B_k = \mathcal{O}(1/k^2)$, then $f(x_k) - f(x_*) = \mathcal{O}(1/k^2)$, which is the rate achieved by Nesterov's optimal method for general smooth convex (but not necessarily strongly convex) functions in the noiseless setting [25, §2.1]. Theorem 2.2 also allows for the possibility that the bound B_k does not converge to zero (necessarily implying the limit of B_{k+1}/B_k is one). In this case, the result simply states that the distance to optimality is eventually bounded by a constant times B_k.

The above analysis allows for the possibility that the approximate gradient is computed by a stochastic algorithm where the error made by the algorithm can be bounded in an absolute sense. We now consider the more general case where the error can only be bounded in expectation. The following result is the counterpart to Theorem 2.2, where we instead have a bound on the expected value of $\|e_k\|^2$.

Theorem 2.3 (Weak expected convergence rates under expected error bounds). *Suppose that* $\mathbb{E}[\|e_k\|^2] \le B_k$, *where*

$$\lim_{k\to\infty} B_{k+1}/B_k \le 1.$$

Then at each iteration of algorithm (1.5) *with* $\alpha_k \equiv 1/L$,

$$\mathbb{E}[f(x_k) - f(x_*)] = \mathcal{O}(C_k),$$

where $C_k = \max\{B_k, (1-\mu/L+\epsilon)^k\}$ *for any* $\epsilon > 0$ *such that* $\epsilon < \mu/L$.

Proof. We use Lemma 2.1 and take expectations of (2.1) to obtain

$$\mathbb{E}[f(x_{k+1}) - f(x_*)] \le (1-\mu/L)\mathbb{E}[f(x_k) - f(x_*)] + \frac{1}{2L}\mathbb{E}[\|e_k\|^2].$$

Proceeding as in the proof of Theorem 2.2, we obtain a similar result, but based on the expected value of the objective. □

2.2. Strong linear convergence. We now describe a particular construction of the sequence $\{B_k\}$ that allows us to achieve a linear decrease of the function values that applies to every iteration k. This strong guarantee, however, comes at the price of requiring bounds on three quantities that are unknowable for general problems: the strong convexity constant μ, the gradient's Lipschitz constant L, and a non-trivial lower bound on the current iterate's distance to optimality $f(x_k) - f(x_*)$.

In particular, we consider any sequence $\{B_k\}$ that satisfies

$$0 \le B_k \le 2L(\mu/L - \rho)\pi_k, \quad k = 1, 2, \dots, \tag{2.8}$$

where $\rho \le \mu/L$ is a positive constant, which controls the convergence rate, and π_k is a non-negative lower bound on the distance to optimality, i.e.,

$$0 \le \pi_k \le f(x_k) - f(x_*). \tag{2.9}$$

Theorem 2.4 (Strong linear rate under absolute error bounds). *Suppose that* $\|e_k\|^2 \le B_k$ *where* B_k *is given by* (2.8). *Then at each iteration of algorithm* (1.5) *with* $\alpha_k \equiv 1/L$,

$$f(x_{k+1}) - f(x_*) \le (1-\rho)[f(x_k) - f(x_*)].$$

Proof. Because $\|e_k\|^2 \le B_k$, Lemma 2.1 and (2.9) imply that

$$\begin{aligned} f(x_{k+1}) - f(x_*) &\le (1-\mu/L)[f(x_k) - f(x_*)] + \frac{1}{2L}B_k \\ &\le (1-\mu/L)[f(x_k) - f(x_*)] + (\mu/L - \rho)\pi_k \\ &\le (1-\mu/L)[f(x_k) - f(x_*)] + (\mu/L - \rho)[f(x_k) - f(x_*)] \\ &= (1-\rho)[f(x_k) - f(x_*)], \end{aligned}$$

as required. □

As we did with Theorem 2.3, we consider the case where the approximate gradient is computed by a stochastic algorithm, and the error can only be bounded in expectation. Provided we now have a lower bound π_k on the expected sub-optimality, i.e.,

$$0 \le \pi_k \le \mathbb{E}[f(x_k) - f(x_*)],$$

it is possible to show an expected linear convergence rate that parallels Theorem 2.4. The proof follows that of Theorem 2.4, where we instead begin by taking expectation of both sides of (2.1).

Theorem 2.5 (Strong expected linear rate under expected error bounds). *Suppose that* $\mathbb{E}[\|e_k\|^2] \le B_k$ *where* B_k *is given by* (2.8). *Then at each iteration of algorithm* (1.5) *with* $\alpha_k \equiv 1/L$,

$$\mathbb{E}[f(x_{k+1}) - f(x_*)] \le (1-\rho)\mathbb{E}[f(x_k) - f(x_*)].$$

In both scenarios, we obtain the fastest convergence rate in the extreme case where $\rho = \mu/L$. From (2.8), this means that B_k must be zero, i.e., the gradient is exact, and we obtain the classic strong-linear convergence result with error constant $\sigma = 1 - \mu/L$ stated in Section 1.3. However, if we take any positive ρ less than μ/L, then we obtain a slower linear convergence rate but (2.8) allows B_k to be non-zero (as long as $\pi_k > 0$).

The bound (2.8) on the error depends on both the conditioning of the problem (as determined by the parameter μ and L), and on the lower bound on the distance to the optimal function value π_k. This seems intuitive. For example, we can allow a larger error in the gradient calculation the further the current iterate is from the optimal function value, but a more accurate calculation is needed to maintain the strong linear convergence rate as the iterates approach the solution. Similarly, if the problem is well conditioned so that the ratio μ/L is close to 1, a larger error in the gradient calculation is permitted, but for ill-conditioned problems where μ/L is very small, we require a more accurate gradient calculation.

Note that the analysis of this section holds even if the bounds μ, L, and π_k are not the tightest possible. Unsurprisingly, we obtain the fastest convergence rate when μ is as large and L is as small as possible, while the largest error in the gradient calculation is allowed if π_k is similarly as large as possible. Note that with $\pi_k = 0$, which is a trivial bound that is valid by definition, we require an exact gradient.

3. Application to incremental gradient.

Incremental gradient methods for problem (1.1) are based on the iteration scheme (1.5) with the gradient approximation

$$g(x_k) := \frac{1}{|\mathcal{B}_k|} \sum_{i \in \mathcal{B}_k} \nabla f_i(x_k), \tag{3.1}$$

where the set $\mathcal{B}^k \subseteq \{1, \ldots, n\}$ represents a sample (of at least one) of the gradients that constitute the full gradient $\nabla f(x_k)$. Typically, $\mathcal{B}_k$ contains a single element that is chosen in either a cyclic fashion or sampled at random, and as discussed in §1.3, the convergence rate of the method is sub-linear. As the batch size increases, however, the error in the sampled gradient $g(x_k)$ decreases, and so the batch size can be used to implicitly control the error in the gradient. We use the results of §2 to develop an increasing-batch strategy that improves on this sub-linear rate.

Let $\mathcal{N}_k$ denote the complement of $\mathcal{B}_k$, so that $\mathcal{B}_k \cup \mathcal{N}_k = \{1, \ldots, n\}$. Then the gradient residual, defined by (1.6), satisfies

$$e_k = \frac{M - |\mathcal{B}_k|}{M|\mathcal{B}_k|} \sum_{i \in \mathcal{B}_k} \nabla f_i(x_k) - \frac{1}{M} \sum_{i \in \mathcal{N}_k} \nabla f_i(x_k). \tag{3.2}$$

The first term is a re-weighting of the gradient approximation and the second term is the portion of $\nabla f(x_k)$ that is not sampled. Assumption (1.7) allows us to bound the norm of this residual in terms of the full gradient, and thus leverage the results of §2.

3.1. Deterministic error bounds.

It follows from (1.7) and (3.2) that

$$\begin{aligned} \|e_k\|^2 &= \left\| \left(\frac{M - |\mathcal{B}_k|}{M|\mathcal{B}_k|} \right) \sum_{i \in \mathcal{B}_k} \nabla f_i(x_k) - \frac{1}{M} \sum_{i \in \mathcal{N}_k} \nabla f_i(x_k) \right\|^2 \\ &\leq 4 \left(\frac{M - |\mathcal{B}_k|}{M} \right)^2 (\beta_1 + \beta_2 \|\nabla f(x_k)\|^2), \end{aligned}$$

where the triangle-inequality is applied repeatedly, and the resulting terms simplified. Next, in the same way that (2.4) was derived, we use (2.2a) to derive the upper bound

$$\|\nabla f(x_k)\|^2 \le 2L[f(x_k) - f(x_*)].$$

Thus the bound on $\|e_k\|^2$ can be expressed in terms of the batch-size ratio and the distance to optimality:

$$\|e_k\|^2 \le 4\left(\frac{M - |\mathcal{B}_k|}{M}\right)^2 \bigl(\beta_1 + 2\beta_2 L[f(x_k) - f(x_*)]\bigr). \tag{3.3}$$

The following result parallels Theorem 2.2, and asserts that a linearly increasing batch size is sufficient to induce a weak linear convergence rate of the algorithm.

Theorem 3.1 (Weak linear rate with increasing batch size). *Suppose that* (1.7) *holds, and that the batch size $|\mathcal{B}_k|$ increases linearly towards M, i.e.,*

$$\rho_k := (M - |\mathcal{B}_k|)/M = \mathcal{O}(\gamma^{k/2})$$

for some $\gamma < 1$. Then at each iteration of algorithm (1.5) *with $\alpha_k \equiv 1/L$,*

$$f(x_k) - f(x_*) = \mathcal{O}(\sigma^k) \tag{3.4}$$

where $\sigma = \max\{\gamma,\, 1 - \mu/L\} + \epsilon$ for any $\epsilon > 0$ such that $\epsilon < \mu/L$.

Proof. Using (3.3) and Lemma 2.1, we obtain the bound

$$\begin{aligned} f(x_{k+1}) - f(x_*) &\le (1 - \mu/L)[f(x_k) - f(x_*)] + \frac{2\rho_k^2}{L}(\beta_1 + 2\beta_2 L[f(x_k) - f(x_*)]) \\ &= (1 - \mu/L + 4\beta_2\rho_k^2)[f(x_k) - f(x_*)] + \frac{2\beta_1}{L}\rho_k^2 \\ &= \omega_k[f(x_k) - f(x_*)] + \frac{2\beta_1}{L}\rho_k^2, \end{aligned}$$

where $\omega_k := 1 - \mu/L + 4\beta_2\rho_k^2$. Applying this recursively, we obtain

$$f(x_k) - f(x_*) \le \mathcal{O}([w_k]^k) + \sum_{i=0}^{k-1} \mathcal{O}([\omega_k]^{k-i-1}\gamma^i).$$

We now take $\delta_k := \max\{\gamma,\, \omega_k\}$ and obtain

$$f(x_k) - f(x_*) \le \mathcal{O}([k+1][\delta_k]^{k-1}).$$

To construct the bounding sequence $\{\xi\sigma^k\}$, first note that because $\rho_k \to 0$, it follows that $\delta_k \to \bar{\delta} := \max\{\gamma,\, 1 - \mu/L\} < 1$. Choose any $\sigma \in (\bar{\delta}, 1)$. Thus, $\sigma > \delta_k$ for all k large enough and we subsequently choose $\xi \ge ([k+1]/\delta_k)(\delta_k/\sigma)^k$, which is possible because the maximum of the right-hand side exists. □

Note an importance difference with Theorem 2.2 where we considered a generic error in the gradient: when the batch size is used to control the error in the gradient, the error in the objective function decreases at twice the rate that the batch size increases.

It is possible to get a strong linear rate of convergence by increasing the batch size in a more controlled way. Theorem 2.4 guides the choice of the batch size, and gives the following corollary. The proof follows by simply ensuring that the right-hand side of (3.3) is bounded as required by Theorem 2.4.

COROLLARY 3.2 (Strong linear rate with increasing batch size). *Suppose that* (1.7) *holds, and that the batch size* $|\mathcal{B}_k|$ *is increased so that at each iteration* $k = 1, 2, \ldots,$

$$4\left(\frac{M - |\mathcal{B}_k|}{M}\right)^2 \left(\beta_1 + 2\beta_2 L[f(x_k) - f(x_*)]\right) \le L(\mu/L - \rho)[f(x_k) - f(x_*)]$$

for some positive $\rho \le \mu/L$. *Then at each iteration of algorithm* (1.5) *with* $\alpha_k \equiv 1/L$,

$$f(x_{k+1}) - f(x_*) \le (1 - \rho)[f(x_k) - f(x_*)].$$

We see that if the individual functions f_i are very similar (so that β_1 and β_2 are small) then we can choose a fairly small batch size. In contrast, if the f_i are very dissimilar then we must use a larger batch size.

3.2. Expected error bounds. Theorem 3.1 and Corollary 3.2 are based on the deterministic bound (3.3) on the gradient error, and hold irrespective of the manner in which the elements of the batches $\mathcal{B}_k$ are chosen, e.g., the batches do not need to be chosen cyclically or sampled uniformly. We can obtain a tighter bound, however, if we choose the batch by uniform sampling (without replacement), which is equivalent to modeling the objective f as

$$f_k(x) = \frac{1}{|\mathcal{B}_k|}\sum_{i=1}^{M} \varepsilon_i f_i(x),$$

where ε_i are identically-distributed Bernoulli random variables with parameter $|\mathcal{B}_k|/M$. In particular, by suitably modifying the derivation of a well-known result in statistical sampling, we can obtain a bound in terms of the quantity

$$S = \frac{1}{M-1}\sum_{i=1}^{M} ||\nabla f_i(x) - \nabla f(x)||^2,$$

which is similar to the sample variance of the gradients. Then, using an argument similar to [15, §2.7],

$$\mathbb{E}[\|e_k\|^2] = \left(\frac{M - |\mathcal{B}_k|}{M}\right)\frac{S}{|\mathcal{B}_k|}.$$

By using (1.7), we obtain the bound

$$\mathbb{E}[\|e_k\|^2] \le \left(\frac{M - |\mathcal{B}_k|}{M}\right)\frac{M}{(M-1)|\mathcal{B}_k|}(\beta_1 + 2(\beta_2 - 1)L[f(x_k) - f(x_*)]) \qquad (3.5)$$

An expected convergence result parallel to Theorem 3.1 can then be obtained, where (3.4) is replaced by

$$\mathbb{E}[f(x_k) - f(x_*)] = \mathcal{O}(\sigma^k).$$

Note, however, that the right-hand side of the bound on $\mathbb{E}[\|e_k\|^2]$ is uniformly better than the bound shown in (3.3). Importantly, it initially decreases to zero at a faster rate as the size of the batch increases. Fig. 3.1 illustrates the difference in the leading coefficients (involving only M and B_k) of (3.3) versus (3.5) as the batch size $\mathcal{B}_k \to M$.

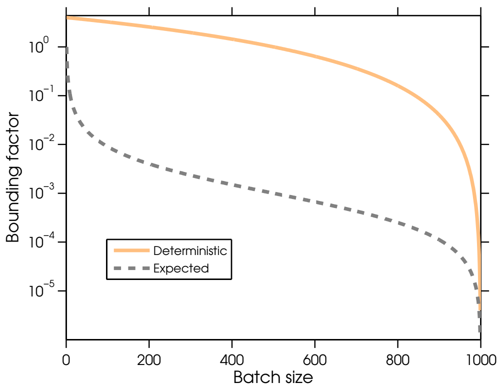

Fig. 3.1. *Bounding factor in the error of the sampled gradient. The expected bound is uniformly better than the deterministic bound.*

4. Practical implementation.

The analysis so far has focused on the basic gradient descent iterations

$$x_{k+1} = x_k - \alpha_k d_k,$$

where $\alpha_k = 1/L$ and $d_k = g(x_k)$. In practice, it is useful to make two modifications to this basic algorithm. The first allows for scaling the direction d_k to account for curvature information in f. The second allows for a varying step size.

4.1. Scaled Direction.

Our implementation attempts to gather useful curvature information on the function f by maintaining a positive-definite approximate Hessian H_k. The search directions d_k are then taken as the solution of the system

$$H_k d = -g(x_k). \tag{4.1}$$

The approximate Hessian is maintained via the recursive application of the update

$$H_{k+1} = U(H_k, y_k, s_k),$$

where U represents an update formula on the kth iteration, and

$$s_k := x_{k+1} - x_k \quad \text{and} \quad y_k := g(x_{k+1}) - g(x_k)$$

measure the change in x and the sampled gradient. In our experiments (see §5) we use a limited-memory BFGS update, which maintains a history of the previous $\ell = 10$ pairs (s_k, y_k), and recursively applies the formula $H_{i+1} = U(H_i, s_i, y_i)$ for $i = k - \ell, \ldots, k$. Nocedal and Wright [27, §9.1] describe the recursive procedure. Updates are skipped if necessary in order to ensure that the Hessian approximation H_k remains positive definite and has a bounded condition number.

If H_k is uniformly positive definite and uniformly bounded in norm (as can be enforced in practice), then assumptions (1.2a)–(1.2b) can be replaced by the following conditions: there exist positive constants L' and μ' such that for all $x, y \in \mathbb{R}^n$ and for all $k = 1, 2, \ldots,$

$$\|\nabla f(x) - \nabla f(y)\|_{H_k^{-1}} \le L' \|x - y\|_{H_k} \tag{4.2a}$$

$$f(y) \ge f(x) + (y - x)^T \nabla f(x) + \tfrac{1}{2}\mu' \|x - y\|_{H_k}^2, \tag{4.2b}$$

where the quadratic norm $\|x\|_{H_k} = \sqrt{x^T H_k x}$ and its dual $\|x\|_{H_k^{-1}}$ are used instead of the Euclidean norm. It can then be verified that all of the preceding results in §§2–3 then apply to the Newton-like algorithm

$$x_{k+1} = x_k - \alpha_k d_k,$$

where d solves (4.1), and the parameters L and μ in those results are replaced by the parameters L' and μ' found in (4.2). The benefit of this approach is that a judicious choice of the scaling H_k can lead to a scaled condition number L'/μ' that can be smaller than the condition number L/μ of the unscaled objective f, effectively improving on the error constants found in the earlier convergence results.

4.2. Varying stepsize. A weakness of our convergence analysis is the requirement for a fixed steplength $\alpha_k \equiv L$, in part because the Lipschitz constant is not usually known, and in part because a dynamic steplength is typically more effective in practice. But a linesearch procedure that ensures a sufficient decrease condition in the true function $f(x)$ runs contrary to a sampling scheme specifically designed to avoid expensive evaluations with f.

In our implementation we attempt to strike a balance between a rigorous linesearch and none at all by enforcing an Armijo-type descent condition on the currently sampled portion of f. In particular, if $\mathcal{B}_k$ is the batch used at iteration k that defines the sampled objective $\bar{f}_k = (1/|\mathcal{B}_k|)\sum_{i\in\beta_k} f_i(x_k)$ and sampled gradient $g(x_k)$, we use a linesearch procedure to select a steplength α_k that satisfies

$$\bar{f}(x_k + \alpha d_k) < \bar{f}(x_k) + \mu\alpha g(x_k)^T d_k, \tag{4.3}$$

where d_k is the current search direction and $\mu \in (0, 1)$. While in deterministic quasi-Newton methods we typically first test whether $\alpha = 1$ satisfies this condition, in our implementation we set our initial trial step length to $\alpha = |\mathcal{B}_{k-1}|/|\mathcal{B}_k|$.

In general $\mathcal{B}_k$ represents only a fraction of all observations, and so the above procedure may not even yield a decrease in the true objective at each iteration. But because we steadily increase the batch size, the procedure just described eventually reduces to a conventional linesearch based on the true objective function f. Hence, the method may initially be nonmonotic, but is guaranteed to be eventually monotonic. Coupled with the choice of search direction described in §4.1, the overall algorithm reduces to a conventional linesearch method with a quasi-Newton Hessian approximation, and inherits the global and local convergence guarantees of that method.

5. Numerical experiments.

This section summarizes a series of numerical experiments in which we apply our incremental gradient method with growing batch size to a series of data fitting applications. Table 5 summarizes the test problems.

The first four experiments are data-fitting applications of logistic regression of varying complexity: binary, multinomial, chain-structured conditional random fields (CRFs), and general CRFs. These logistic-regression applications follow a standard pattern. We first model the probability of an outcome b_i by some log-concave function $p(b_i \mid a_i, x)$, where a_i is data; the goal is to choose the parameters x so that the likelihood function

$$\mathcal{L}(x) = \prod_{i=1}^{M} p(b_i \mid a_i, x)$$

TABLE 5.1
The test problems.

Problem	M	n	Description
binary logistic regression	92,189	823,470	spam identification (§5.1)
multinormal logistic regression	70,000	785	digit identification (§5.2)
chain-structured CRF	8936	1,643,004	noun-phrase chunking (§5.3)
general CRF	50	4	image denoising (§5.4)
nonlinear least squares	101	10,201	seismic inversion (§5.5)

is maximized. (This formulation assumes the pairs (a_i, b_i) are independent.) We then approximate x by minimizing the 2-norm regularized negative log-likelihood function

$$f(x) = \sum_{i=1}^{M} f_i(x) + \tfrac{1}{2}\lambda\|x\|^2 \quad \text{with} \quad f_i(x) = -\log p(b_i \mid a_i, x) \tag{5.1}$$

for some positive regularization parameter λ. These objectives are all strongly convex and satisfy our assumptions in §1.1.

The last experiment is a more general application of nonlinear least-squares to seismic inversion. This last data-fitting application does not satisfy our central convexity assumption, but nonetheless illustrates the practical relevance of our approach on difficult problems.

Our numerical experiments compare the following three methods:

Deterministic. A conventional quasi-Newton linesearch method that uses the true function f and gradient ∇f at every iteration. The method is based on a limited-memory BFGS Hessian approximation and a linesearch based on Hermite cubic-polynomial interpolation and the strong Wolfe conditions. Several comparison studies indicate that these type of limited-memory quasi-Newton methods are among the most efficient deterministic methods available for solving large-scale logistic regression and conditional random field problems [18, 34, 20, 31]

Stochastic. An incremental gradient method based on the iteration (1.5), where at each iteration α_k is held constant and $g(x_k) = \nabla f_i(x_k)$, where the index $i \in 1, \dots, M$ is randomly selected. This corresponds to a constant batch size of one, and this simple method has proved competitive with more advanced deterministic methods like the one above for estimation in CRF models [33].

Hybrid. This is the proposed method described in §4, which uses search directions computed from (4.1) and a linesearch based on satisfying condition (4.3). As with the deterministic method described above, the Hessian approximations are based on limited-memory BFGS and the linesearch uses polynomial interpolation. The gradient approximation $g(x_k)$ is based on (3.1), where $\mathcal{B}_k \subseteq \{1, \dots, M\}$, and the number of elements in the current batch is initially 1, and grows linearly as per the formula

$$|\mathcal{B}_{k+1}| = \lceil \min\{1.1 \cdot |\mathcal{B}_k| + 1,\ M\} \rceil.$$

The hybrid nature of this approach should now be clear: the very first iteration is similar to the stochastic method; when the batch size grows to include all observations, the algorithm morphs into the deterministic method.

All experiments are carried out using Matlab R2010b on a 64-bit Athlon machine. Two plots are shown for each experiment. The first shows the progress of the objective value against the index $p = \frac{1}{M}\sum_{k=1}^{p} |\mathcal{B}_k|$ for $p = 1, 2, \dots$, which measures the effective

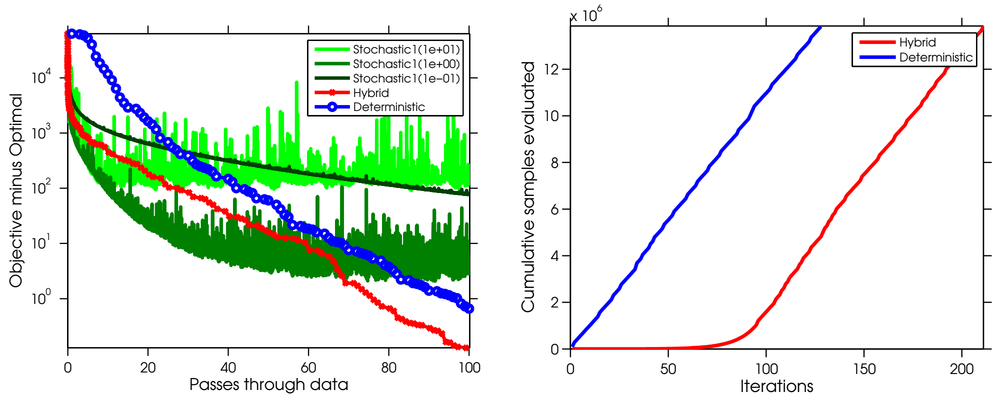

Fig. 5.1. *Binary logistic regression experiments for different optimization strategies for spam classification. The stochastic method is run with 3 different fixed steplengths.*

number of passes through the entire data set. The second plot shows the cumulative number of sample functions evaluated, i.e., $\sum_{i=1}^{k} |\mathcal{B}_k|$, against the iterations k.

5.1. Binary logistic regression. Logistic regression models [5, §4.3.2] are used in an enormous number of applications for the problem of *binary classification*. We are given data with M examples of input-output pairs (a_i, b_i), where $a_i \in \mathbb{R}^n$ is a vector of n features, and $b_i \in \{-1, 1\}$ is a corresponding binary outcome. The goal is to build a linear classifier that, given the features a_i and a vector of parameters x, the sign of the inner-product $a_i^T x$ gives b_i. The logistic model gives the probability that b_i takes the value 1:

$$p_1(b_i = 1 \mid a_i, x) = \frac{\exp(a_i^T x)}{\exp(a_i^T x) + 1} = \frac{1}{1 + \exp(-a_i^T x)}.$$

(Typically a bias variable is added, but we equivalently assume that the first element of x is set to one.) Thus, the probability that b_i takes the value -1 is $[1 - p(b_i = 1 \mid a_i, x)]$. We can write these two cases compactly as

$$p_1(b_i \mid a_i, x) = \frac{1}{1 + \exp(-b_i a_i^T x)}.$$

This is the probability function p used in (5.1).

The dominant cost in computing f and its gradient is the cost of forming the matrix-vector products Ax and $A^T y$ (for some y), where the M rows of the matrix A are formed from the vectors a_i. The Hessian is $\nabla^2 f(x) = A^T D A$, where D is a diagonal with elements $p_1(b_i \mid a_i, x) \cdot [1 - p_1(b_i \mid a_i, x)]$, which lie in the range $(0, 0.25]$. The (nonnegative) eigenvalues of the Hessian are thus bounded above by $0.25\|A\|^2$, and are strictly positive if A has full rank. Combined with 2-norm regularization, the resulting function f satisfies the assumptions of §1.1.

Our experiments for binary logistic regression are based on the TREC 2005 data set, which contains 823,470 binary variables describing the presence of word tokens in 92,189 email messages involved in the legal investigations of the Enron corporation [8]. The target variable indicates whether the email was spam or not. The data set was prepared by Carbonetto [7, §2.6.5], and we set the regularization parameter $\lambda = 0.01$.

The results of this experiment are plotted in Figure 5.1, where we define the optimal value as the best value found across the methods after 150 effective passes

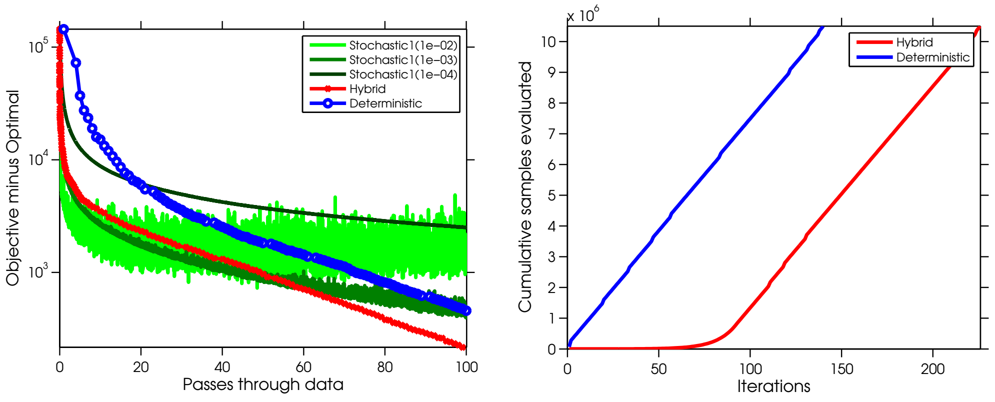

FIG. 5.2. *Multinomial logistic regression experiments for different optimization strategies for digit classification.*

through the data, and where for the stochastic method we plot the three step sizes (among power of 10) that gave the best performance over the time-frame. Of course, in practice we will not know what step size optimizes the performance of the stochastic method, and the lack of sensitivity of the result to the initial step size is an advantage of the deterministic and hybrid methods. In the plot we see that, like the stochastic methods, the hybrid method makes rapid initial progress. Unlike the stochastic method, however, the hybrid method continues to make steady progress similar to the deterministic method. This behavior is in agreement with what the theory predicts.

5.2. Multinomial logistic regression. Multinomial logistic regression relaxes the binary requirement, and allows each outcome b_i to take any value from a set of classes $\mathcal{C}$ [5, §4.3.4]. In this model there is a separate parameter vector x_j for each class $j \in \mathcal{C}$. We model the probability that b_i is assigned a particular class j as

$$p_2(b_i = j \mid a_i, \{x_j\}_{j\in\mathcal{C}}) = \frac{\exp(x_j^T a_i)}{\sum_{j'\in\mathcal{C}} \exp(x_{j'}^T a_i)}. \tag{5.2}$$

This model is equivalent to binary logistic regression in the special case where the parameters x_j of one class are fixed at zero and there are only two classes, i.e., $\mathcal{C} = \{-1, 1\}$. As with binary logistic regression, the function p_2 is log-concave and $-\log p_2$ has a Hessian whose eigenvalues are bounded above. Hence, the resulting function f in (5.1) satisfies the assumptions of §1.1.

Our experiments for multinomial logistic regression are based on the well-known MNIST data set [14], containing 70,000 examples of 28-by-28 images of digits, where each digit is classified as one of the numbers 0 through 9. The results are plotted in Figure 5.2, with $\lambda = 1$. The trends are similar to the binary logistic regression experiments.

5.3. Chain-structured conditional random fields. The binary and multinomial logistic regression models consider the case where a particular outcome b_i is associated with a particular feature vector a_i. The CRF model takes multinomial logistic regression a step further by considering a set of feature vectors a_i^k with which to predict corresponding values for discrete variables $b_i^k \in \mathcal{C}$, where k takes values in a discrete ordered set Ω.

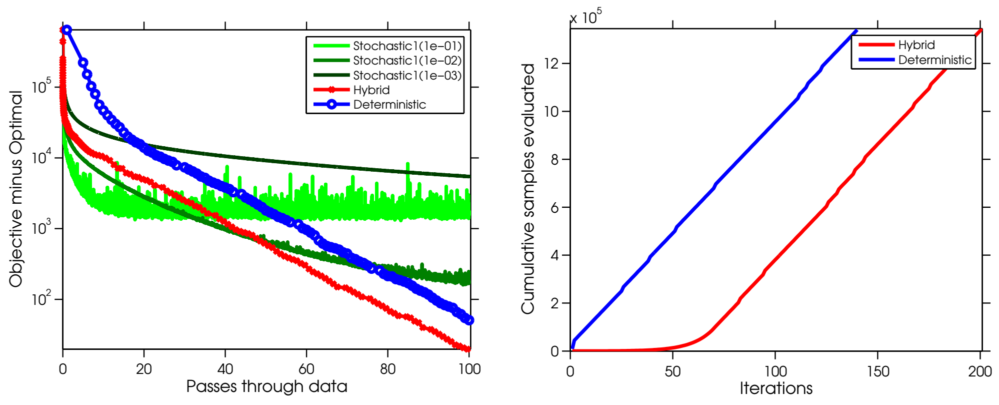

FIG. 5.3. *Chain-structured conditional random field experiments for different optimization strategies on the noun-phrase chunking task.*

We can naturally extend the multinomial logistic model to this scenario by defining the probability of a joint assignment b_i as

$$p(\{b_i^k = j_k\}_{k\in\Omega} \mid \{a_i^k\}_{k\in\Omega}, \{x_j\}_{j\in\mathcal{C}}) = \frac{1}{Z_i}\prod_{k\in\Omega}\exp(x_{j_k}^T a_i^k).$$

(We assume that the parameter vectors x_{j_k} are tied, so that x_{j_k} is constant for all k; this assumption is not required in general.) The normalizing constant

$$Z_i = \sum_{j_1\in\mathcal{C}}\sum_{j_2\in\mathcal{C}}\cdots\sum_{j_k\in\mathcal{C}}\prod_{k\in\Omega}\exp(x_{j_k}^T a_i^k)$$

is chosen so that the distribution sums to one over all possible configurations of the b_i^k variables. As written, computing Z_i involves a very large number of terms, $|\mathcal{C}|^{|\Omega|}$; still, the sum can be computed efficiently by exchanging the order of operations. While this model is a straight-forward generalization of multinomial logistic regression, it assumes that the labels in Ω that we are simultaneously predicting are independent. This might be unrealistic if, for example, the variables come from time-series data where b_i^k and b_i^{k+1} are likely to be correlated.

A chain-structured CRF [13] augments the model with additional terms that take into account sequential dependencies in the labels. It allows pairwise features $a_i^{kk'}$ and associated parameters $x_{kk'}$, and defines the probability of a configuration as

$$\begin{aligned} p_3(\{b_i^k = j\}_{k\in\Omega} \mid \{a_i^k\}_{k\in\Omega}, \{a_i^{kk'}\}_{k,k'\in\Omega,k'=k+1}, \{x_j\}_{j\in\mathcal{C}}, \{x_{j,j'}\}_{j,j'\in\mathcal{C}}) \\ = \frac{1}{Z_i}\left[\prod_{k\in\Omega}\exp(x_{j_k}^T a_i^k)\right]\cdot\left[\prod_{\substack{k,k'\in\Omega\\ k'=k+1}}\exp(x_{j_k j_{k'}}^T a_i^{kk'})\right]. \end{aligned}$$

The normalizing constant Z_i is again set so that the distribution sums to one, and it can be computed using a variant on the forward-backward algorithm used in hidden Markov models [29, § III]. Because we need to run the forward-backward algorithm for each i, the probability function p_3 is significantly more expensive to evaluate than the corresponding multinomial probability function p_2; see (5.2).

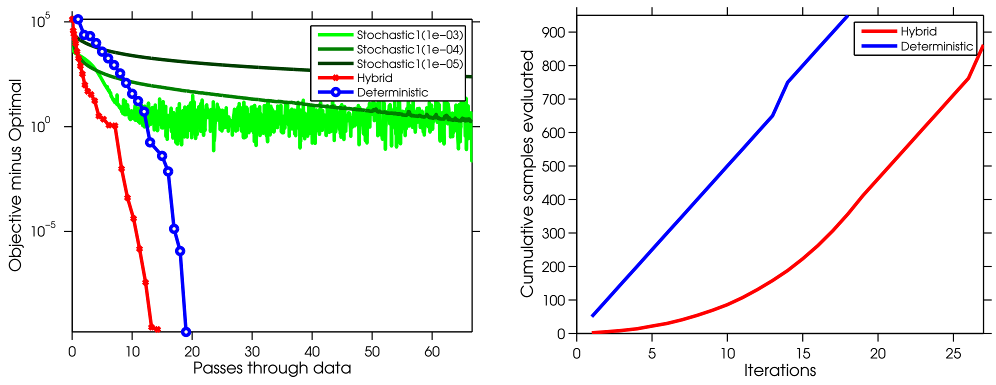

FIG. 5.4. *Lattice-structured conditional random field experiments for different optimization strategies for an image denoising task.*

For our chain-structured CRF experiments, we use the noun-phrase chunking problem from the CoNLL-2000 Shared Task [30], where the goal is to assign each word in a sentence to one of 22 possible states, and we use approximately 1.6 million features to represent the presence of words and word combinations [30, 31]. The CoNLL-2000 data set contains 211,727 words grouped into 8,936 sentences. The results of this experiment are plotted in Figure 5.3, where the regularization parameter is $\lambda = 1$.

5.4. General conditional random fields. While chain-structures can model sequential dependencies in the labels, we might be interested in other structures, such as lattice structures for modeling image data. In order to use general CRFs [11, §4.6,1 and §20.3.2], we define a graph $\mathcal{G}$ where the nodes are the labels $\{1, 2, \dots, w\}$ and the edges are the variable dependencies that we wish to consider. We then define the probability of b_i taking a configuration $b_i^k = j_k$, for $k \in \Omega$, as

$$p_4(\{b_i^k = j_k\}_{k\in\Omega} \mid \{a_i^{k'}\}_{k'\in\Omega}, \{a_i^{kk'}\}_{(k,k')\in\mathcal{E}}, \{x_j\}_{j\in\mathcal{C}}, \{x_{jj'}\}_{(j,j')\in\mathcal{C}}) \\ = \frac{1}{Z_i}\left[\prod_{k\in\Omega} \exp(x_{j_k}^T a_i^k)\right] \cdot \left[\prod_{(k,k')\in\mathcal{E}} \exp(x_{j_k j_{k'}}^T a_i^{kk'})\right].$$

In this general case, computing Z_i is in the complexity class $\sharp P$, and the best known algorithms have a runtime that is exponential in the tree-width of the graph $\mathcal{G}$ [11, §9-10]. For a two-dimensional lattice structure, the tree-width is the minimum between the two dimensions of the structure, so computing Z_i is only feasible if one of the dimensions is very small (in the degenerate one-dimensional chain-structured case, the tree-width is one).

Because of the intractability of computing Z_i, we consider optimizing a pseudo-likelihood approximation [4] based on the probability model

$$p_4(\{b_i^k = j_k\}_{k\in\Omega} \mid \{a_i^k\}_{k\in\Omega}, \{a_i^{kk'}\}_{(k,k')\in\mathcal{E}}, \{x_j\}_{j\in\mathcal{C}}, \{x_{jj'}\}_{(j,j')\in\mathcal{C}}) \\ \approx \prod_{k\in\Omega} p(b_i^k = j_k \mid \{a_i^{k'}\}_{k'\in\Omega}, \{a_i^{kk'}\}_{(k,k')\in\mathcal{E}}, \{x_j\}_{j\in\mathcal{C}}, \{x_{jj'}\}_{(j,j')\in\mathcal{C}}, \{b_i^{k'}\}_{k'\in\Omega, k'\neq k}).$$

The individual terms in this product of conditionals have the form of a multinomial logistic regression probability and are straightforward to compute. This is the function p used to define the objective function f in (5.1).

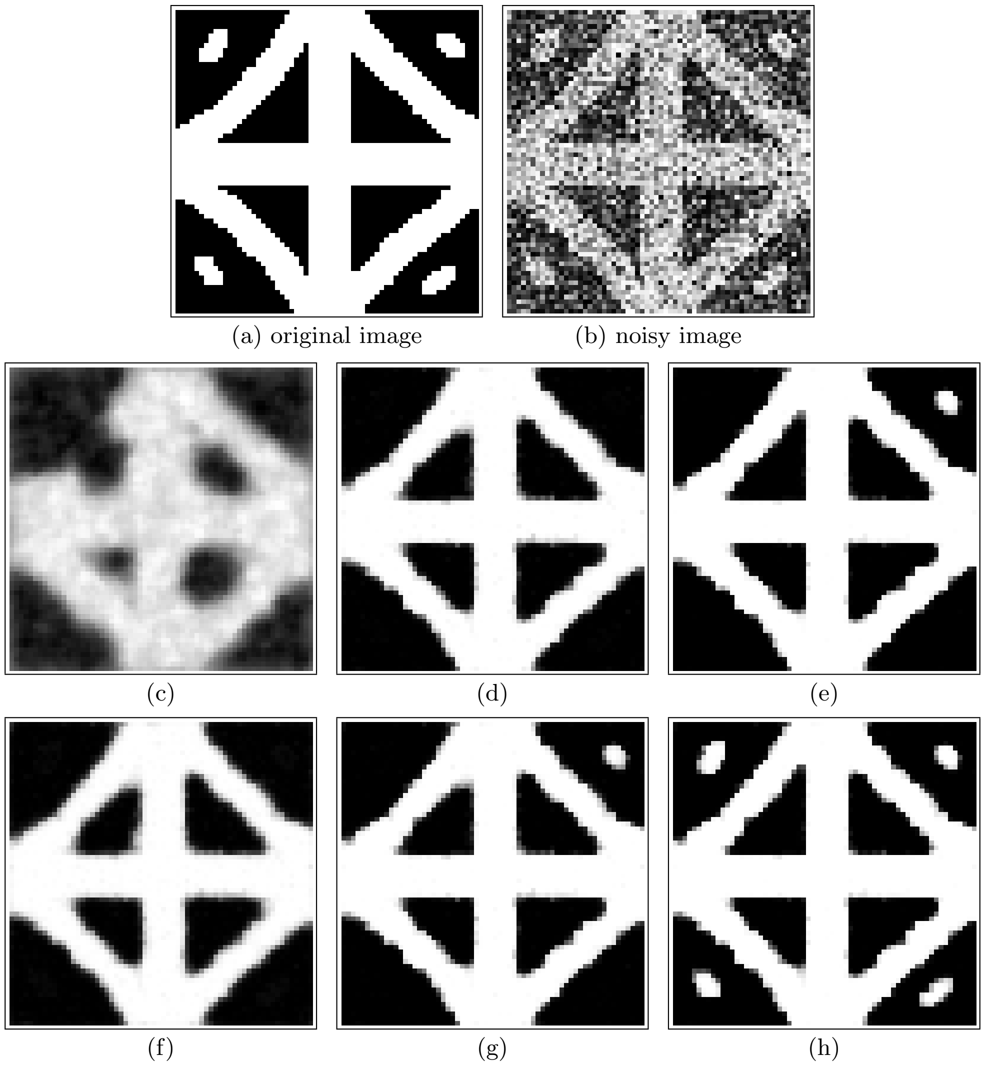

Fig. 5.5. *Top row: original (a) and noisy (b) image. Second row: marginals after 2 passes through the data for deterministic (c), stochastic (d), and hybrid (e). Third row: marginals after 5 passes through the data for deterministic (f), stochastic (g), and hybrid (h).*

Our experiments on general CRFs are based on the image-denoising experiments described by Kumar and Hebert [12]. We use their set of 50 synthetic 64-by-64 images. Figure 5.4 shows the performance of the different methods with a regularization parameter of $\lambda = 1$. Figure 5.5 illustrates the marginal probabilities for the different methods at various points in the optimization for a randomly-chosen image in the data set. (For the stochastic method, we plot the result with a step size of $\alpha = 10^{-4}$.) To approximate these marginals, we use the loopy belief propagation message-passing algorithm [5, §8.4.7]. In these plots we see that the deterministic method does poorly even after two full passes through the data set, while the stochastic and hybrid methods do much better. After five passes through the data set, the hybrid method has found a solution that is visually nearly indistinguishable from the true solution, while it is

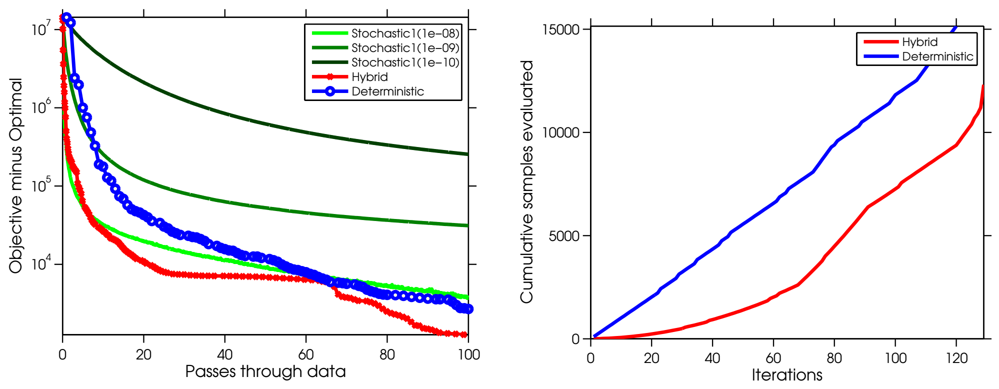

FIG. 5.6. *Nonlinear least-squares experiments for different optimization strategies on a seismic inversion problem.*

still possible to see obvious differences in the deterministic and stochastic methods.

5.5. Seismic inversion. This last numerical experiment is a seismic inversion problem described by van Leeuwen et al. [32]. The aim here is to recover an image of underground geological structures using only data collected by geophone receivers placed at the surface of the earth; these geophones record acoustic "shots" created by sources also at the surface.

The waveform inversion problem attempts to find a model m of the subsurface structure that minimizes the nonlinear least-squares misfit as measured by the function

$$\phi(x) = \sum_{i=1}^{M} \sum_{\omega \in \Omega} \|d_i - PH_\omega(x)^{-1} q_i\|^2.$$

Each index i corresponds to a particular shot (i.e., an observation) created by the source q_i which creates a measurement d_i; each experiment samples a set of frequencies $\omega \in \Omega$. The matrix P accounts for each receiver location. The main cost in evaluating ϕ is solving the Helmholtz equation $H_\omega[x]u = q_i$ for each i, which is an expensive nonlinear partial differential equation. Regularization is typically achieved by truncating the solution process [32]. Although the function ϕ is nonconvex and does not satisfy our standing assumptions, it hints at the applicability of the hybrid approach for solving difficult problems of important practical interest.

The results shown in Figure 5.6 are based on a relatively small 2-dimensional example that involves M=101 sources measuring 8 frequencies. (Larger experiments, especially in 3-dimensions, are only feasibly accomplished on a computing cluster.) Because the problem is not strongly convex and hence we do not expect a fast convergence rate when far from this solution, for this problem the hybrid method increases the batch size by only one sample at each iteration, i.e., $|\mathcal{B}_{k+1}| = \min\{|\mathcal{B}_k| + 1, M\}$. After 20 passes through the data, the hybrid method clearly outperforms the deterministic method. The best stochastic method performs nearly as well as the deterministic and hybrid methods for the first 60 iterations, though with other step sizes it performs poorly. Although the figure shows the best methods all achieving similar residuals after 60 passes through the data, in practice that involves a prohibitive number of Helmholtz solves; practitioners are interested in making quick progress with as few solves as possible (and without needing to test a variety of step sizes to achieve good performance).

6. Discussion. Our work has focused on inexact gradient methods for the unconstrained optimization of differentiable strongly-convex objectives. We anticipate that a similar convergence analysis with inexact gradients could be applied to other algorithms, such as Nesterov's accelerated gradient method [25, §2.1]. In this case, it may be possible to relax the strong convexity assumption, and obtain the optimal $\mathcal{O}(1/k^2)$ rate for an inexact-gradient version of this algorithm. The optimal $\mathcal{O}(1/k^2)$ rate using a gradient approximation whose error is uniformly bounded across iterations has been established by several authors, notably d'Aspremont [9]. But allowing a variable error would encourage more flexibility in early the iterations, and would allow for eventually solving the problem to arbitrary accuracy. Although the $\mathcal{O}(1/k^2)$ rate is sub-linear, it is substantially faster than the optimal $\mathcal{O}(1/\sqrt{k})$ achievable by methods that use noisy gradient information [23, §14.1].

We might also obtain analogous rates for proximal-gradient methods for optimization with convex constraints or non-differentiable composite optimization problems, such as 1-norm regularization [26]. The more general class of mirror-descent methods [1], which are useful for problems with a certain geometry such as optimization with simplex constraints, also seem amenable to analysis in our controlled error scenario.

We considered the case of bounded noise or noise that can be bounded in expectation, and subsequently derived convergence rates and expected convergence rates, respectively. A third scenario that could be analyzed is the case where the noise is bounded with a certain probability. If the individual $\nabla f_i(x)$ are concentrated around $\nabla f(x)$, this might allow us to use concentration inequalities [19] to show that the convergence rates hold with high probability. Although we have analyzed an arbitrary strategy for selecting the elements of the batch, and shown that uniform sampling achieves a better bound in expectation, it is possible that a quasi-random selection of the individual gradients might further refine the bound [21].

Although our emphasis here is on data fitting applications where the error is a by-product of subsampling the data, our analysis and implementation may be useful for other problems. For example, Gill et al. [10, p. 357] discuss the case of an objective function that can be evaluated to a prescribed accuracy (e.g., it could depend on an iterative process or a discretization level). They suggest solving the optimization problem over a sequence of tighter function accuracies. Our work provides a formal analysis and practical implementation of a method where the accuracy might be increased dynamically each iteration rather than solving a sequence of intermediate optimization problems. As a more recent example, [28] consider approximating gradients in non-Gaussian state-space models using particle filters. Here, the variance of the approximation is directly proportional to the number of particles, and thus our work provides guidelines for selecting the number of particles to use in the approximation on each iteration.

Acknowledgements. We would like to thank Kevin Swersky for the suggestion of the growing-batch strategy, and Sasha Aravkin, Francis Bach, Hoyt Koepke, Pierre Jacob, and Nicholas Le Roux for valuable discussions.

REFERENCES

[1] A. Beck and M. Teboulle, *Mirror descent and nonlinear projected subgradient methods for convex optimization*, Oper. Res. Lett., 31 (2003), pp. 167–175.
[2] D.P. Bertsekas, *A new class of incremental gradient methods for least squares problems*, SIAM J. Optim., 7 (1997), pp. 913–926.
[3] D.P. Bertsekas and J.N. Tsitsiklis, *Neuro-dynamic programming*, Athena Scientific, 1996.

[4] J. Besag, *Statistical analysis of non-lattice data*, The Statistician, 24 (1975), pp. 179–195.
[5] C.M. Bishop, *Pattern recognition and machine learning*, Springer, 2006.
[6] D. Blatt, A.O. Hero, and H. Gauchman, *A convergent incremental gradient method with a constant step size*, SIAM J. Optim., 18 (2008), pp. 29–51.
[7] P. Carbonetto, *New probabilistic inference algorithms that harness the strengths of variational and Monte Carlo methods*, PhD thesis, Univ. of British Columbia, Vancouver, BC, May 2009.
[8] G. V. Cormack and T. R. Lynam, *Spam corpus creation for TREC*, in Proc. 2nd Conference on Email and Anti-Spam, 2005. `http://plg.uwaterloo.ca/~gvcormac/treccorpus/`.
[9] A. d'Aspremont, *Smooth Optimization with Approximate Gradient*, SIAM J. Optim., 19 (2008), pp. 1171–1183.
[10] P.E. Gill, W. Murray, and M.H. Wright, *Practical optimization*, Academic Press, 1981.
[11] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, MIT Press, 2009.
[12] S. Kumar and M. Hebert, *Discriminative fields for modeling spatial dependencies in natural images*, Advances in neural information processing systems, 16 (2004).
[13] J. Lafferty, A. McCallum, and F. Pereira, *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*, Int. C. Mach. Learn., (2001).
[14] Y. LeCun and C. Cortes, *The MNIST database*. `http://yann.lecun.com/exdb/mnist/`, 1998.
[15] S.L. Lohr, *Sampling: design and analysis*, Duxbury Press, 1999.
[16] D.G. Luenberger and Y. Ye, *Linear and nonlinear programming*, Springer Verlag, 2008.
[17] Z.Q. Luo and P. Tseng, *Error bounds and convergence analysis of feasible descent methods: A general approach*, Ann. Oper. Res., 46 (1993), pp. 157–178.
[18] R. Malouf, *A comparison of algorithms for maximum entropy parameter estimation*, Inter. Conf. on Computational Linguistics, (2002).
[19] P. Massart, *Concentration Inequalities and Model Selection*, Lecture Notes in Mathematics, 1896 (2007).
[20] T.P. Minka, *Algorithms for maximum-likelihood logistic regression*, tech. report, CMU, 2003.
[21] W.J. Morokoff and R.E. Caflisch, *Quasi-random sequences and their discrepancies*, SIAM J. on Sci. Comp., 15 (1994), pp. 1251–1279.
[22] A. Nedic and D. Bertsekas, *Convergence rate of incremental subgradient algorithms*, Stochastic Optimization: Algorithms and Applications, (2000), pp. 263–304.
[23] A. Nemirovski, *Efficient methods in convex programming*, Lecture notes, (1994).
[24] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, *Robust stochastic approximation approach to stochastic programming*, SIAM J. Optim., 19 (2009), pp. 1574–1609.
[25] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*, Springer Netherlands, 2004.
[26] ———, *Gradient methods for minimizing composite objective function*, CORE Discussion Paper 76, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2007.
[27] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, second ed., 2006.
[28] G. Poyiadjis, A. Doucet, and S.S. Singh, *Particle approximations of the score and observed information matrix in state space models with application to parameter estimation*, Biometrika, 98 (2011), pp. 65–80.
[29] L.R. Rabiner, *A tutorial on hidden Markov models and selected applications in speech recognition*, Proc. of the IEEE, 77 (1989), pp. 257–286.
[30] E. Sang and S. Buchholz, *Introduction to the CoNLL-2000 Shared Task: Chunking*, in Proc. of the Conf. on Natural Language Learning, Lisbon, Portugal, 2000, pp. 127–132.
[31] F. Sha and F. Pereira, *Shallow parsing with conditional random fields*, Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, (2003).
[32] T. van Leeuwen, Schmidt M, M. P. Friedlander, and F. Herrmann, *A hybrid stochastic-deterministic optimization method for waveform inversion*, in 73rd EAGE Conference & Exhibition, May 2011.
[33] S. V. N Vishwanathan, N.N. Schraudolph, M.W. Schmidt, and K.P. Murphy, *Accelerated training of conditional random fields with stochastic gradient methods*, Int. C. Mach. Learn., (2006), pp. 969–976.
[34] H. Wallach, *Efficient training of conditional random fields*, master's thesis, Univ. of Edinbrugh, 2002.