# A Stochastic Quasi-Newton Method for Online Convex Optimization*

**Nicol N. Schraudolph**
nic.schraudolph@nicta.com.au

**Jin Yu**
jin.yu@rsise.anu.edu.au

**Simon Günter**
simon.guenter@nicta.com.au

Statistical Machine Learning, National ICT Australia
Locked Bag 8001, Canberra ACT 2601, Australia

Research School of Information Sciences & Engineering
Australian National University, Canberra ACT 0200, Australia

## Abstract

We develop stochastic variants of the well-known BFGS quasi-Newton optimization method, in both full and memory-limited (LBFGS) forms, for online optimization of convex functions. The resulting algorithm performs comparably to a well-tuned natural gradient descent but is scalable to very high-dimensional problems. On standard benchmarks in natural language processing, it asymptotically outperforms previous stochastic gradient methods for parameter estimation in conditional random fields. We are working on analyzing the convergence of online (L)BFGS, and extending it to non-convex optimization problems.

## 1 INTRODUCTION

Machine learning poses data-driven optimization problems in which the objective function involves the summation of loss terms over a set of data to be modeled. Classical optimization techniques must compute this sum in its entirety for each evaluation of the objective, respectively its gradient. As available data sets grow ever larger, such "batch" optimizers therefore become increasingly inefficient. They are also ill-suited for the online (incremental) setting, where partial data must be modeled as it arrives.

Stochastic (online) gradient-based methods, by contrast, work with gradient estimates obtained from small subsamples (mini-batches) of training data. This can greatly reduce computational requirements: on large, redundant data sets, simple stochastic gradient descent routinely outperforms sophisticated second-order batch methods by orders of magnitude (*e.g.*, Vishwanathan et al., 2006), in spite of the slow convergence of first-order gradient descent. Schraudolph (1999, 2002) further accelerates stochastic gradient descent through online adaptation of a gain vector.

Attempts to develop more advanced stochastic gradient methods are hampered by the fact that core tools of conventional gradient-based optimization, such as line searches and Krylov subspaces, are not amenable to stochastic approximation (Schraudolph and Graepel, 2003). Online implementations of conjugate gradient methods (Møller, 1993; Schraudolph and Graepel, 2003) have therefore proven largely ineffective.

The most successful online second-order learning algorithms to date perform either system identification by global extended Kalman filtering (Puskorius and Feldkamp, 1991), or natural gradient descent (Amari et al., 2000). Both work by incrementally maintaining an estimate of the covariance of the residuals (respectively gradient), whose inverse is then used to scale the parameter update. While quite effective, these methods do not model the curvature (Hessian) of the loss function, and require $O(n^2)$ space and time per iteration to optimize a system with $n$ parameters.

Here we overcome these limitations by systematically modifying the Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton method, in both its full and memory-limited (LBFGS) variants, so as to make it amenable to stochastic approximation of gradients. This results in a fast, scalable, stochastic quasi-Newton method for online convex optimization that outperforms previous approaches.

We first introduce a simple stochastic model, and consider the performance of previous stochastic gradient methods on it. In Section 3 we briefly review the BFGS and LBFGS algorithms, and discuss the changes required to make them work online. Section 4 evaluates the resulting algorithms against competing methods on a non-realizable stochastic model, and on conditional random field (CRF) parameter estimation for

* *Proc. 11th Intl. Conf. Artificial Intelligence and Statistics*, San Juan, Puerto Rico, 2007. ISBN 0-9727358-2-8.

natural language processing. Section 5 discusses our results and ongoing work.

# 2 PRELIMINARIES

We set up a simple optimization problem to serve as a model illustrating the performance of stochastic gradient methods as they are subsequently introduced.

## 2.1 OBJECTIVE FUNCTION MODEL

We follow Schraudolph and Graepel (2003) in their choice of a simple quadratic (albeit ill-conditioned and semi-sparse) stochastic model problem.

### 2.1.1 Deterministic Quadratic

The $n$-dimensional quadratic provides us with the simplest possible test setting that differentiates between various gradient methods. In its deterministic form, the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is given by

$$f(\boldsymbol{\theta}) = \tfrac{1}{2}\,(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^\top \boldsymbol{J}\boldsymbol{J}^\top\,(\boldsymbol{\theta} - \boldsymbol{\theta}^*)\,, \qquad (1)$$

where $\boldsymbol{\theta}^* \in \mathbb{R}^n$ is the optimal parameter, and $\boldsymbol{J} \in \mathbb{R}^{n\times n}$ the Jacobian matrix, both of our choosing. By definition, the Hessian $\boldsymbol{H} = \boldsymbol{J}\boldsymbol{J}^\top$ is constant and positive semi-definite here; the gradient is $\nabla f(\boldsymbol{\theta}) = \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)$.

### 2.1.2 Stochastic Quadratic

A stochastic optimization problem analogous to the above can be defined by the data-dependent objective

$$f(\boldsymbol{\theta}, \boldsymbol{X}) = \frac{1}{2\,b}\,(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^\top \boldsymbol{J}\boldsymbol{X}\boldsymbol{X}^\top\boldsymbol{J}^\top(\boldsymbol{\theta} - \boldsymbol{\theta}^*)\,, \qquad (2)$$

where $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_b]$ is an $n \times b$ matrix collecting a *batch* of $b$ random input vectors to the system, each drawn i.i.d. from a normal distribution: $\boldsymbol{x}_i \sim N(\mathbf{0}, \boldsymbol{I})$. This means that $\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^\top] = b\,\boldsymbol{I}$, so that in expectation this is identical to the deterministic formulation (1):

$$\mathbb{E}_{\boldsymbol{X}}[f(\boldsymbol{\theta}, \boldsymbol{X})] \;=\; \frac{1}{2\,b}\,(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^\top \boldsymbol{J}\,\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^\top]\boldsymbol{J}^\top(\boldsymbol{\theta} - \boldsymbol{\theta}^*) = f(\boldsymbol{\theta})\,. \qquad (3)$$

The optimization problem is harder here since the objective can only be probed by supplying stochastic inputs to the system, giving rise to the noisy estimates $\boldsymbol{H} = b^{-1}\boldsymbol{J}\boldsymbol{X}\boldsymbol{X}^\top\boldsymbol{J}^\top$ and $\nabla f(\boldsymbol{\theta}, \boldsymbol{X}) = \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)$ of the true Hessian and gradient, respectively. The degree of stochasticity is determined by the batch size $b$; the system becomes deterministic in the limit as $b \to \infty$.

### 2.1.3 Choice of Jacobian

For our experiments we choose the Jacobian $\boldsymbol{J}$ such that the Hessian has a) eigenvalues of widely differing magnitude (ill-conditioning), and b) eigenvectors of intermediate sparsity. We achieve this by imposing some sparsity on the notoriously ill-conditioned Hilbert matrix, defining

$$\boldsymbol{J}_{ij} := \begin{cases} \frac{1}{i+j-1} & \text{if } \; i \bmod j \;=\; 0 \\ & \text{or } j \bmod i \;=\; 0\,, \\ 0 & \text{otherwise}\,. \end{cases} \qquad (4)$$

Like Schraudolph and Graepel (2003) we use unconstrained online minimization of (2), with $\boldsymbol{J}$ given by (4) in $n = 5$ dimensions (condition number of the Hessian: $4.9 \cdot 10^3$), as our model problem for stochastic gradient methods.

## 2.2 STOCHASTIC GRADIENT METHODS

We now briefly review three stochastic gradient optimization algorithms, representative of the spectrum of such methods developed to date, and illustrate their performance on the model problem introduced above.

### 2.2.1 Stochastic Gradient Descent (SGD)

Simple stochastic gradient descent takes the form

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla f(\boldsymbol{\theta}_t, \boldsymbol{X}_t), \qquad (5)$$

where $\boldsymbol{\theta}_t$ is the current parameter estimate, $\eta_t > 0$ a scalar gain, and $\boldsymbol{X}_t$ the current batch of data. Robbins and Monro (1951) have shown that (5) converges to $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$, provided that

$$\sum_t \eta_t = \infty \quad \text{and} \quad \sum_t \eta_t^2 < \infty\,. \qquad (6)$$

A commonly used decay schedule for $\eta_t$ that fulfills these conditions is given by

$$\eta_t = \frac{\tau}{\tau + t}\,\eta_0\,, \qquad (7)$$

where $\eta_0, \tau > 0$ are tuning parameters. We employ this schedule in the experiments of Section 4; for our simple quadratic model a constant gain proved sufficient.

SGD takes only $O(n)$ space and time per iteration. Although it can greatly outperform sophisticated batch methods on large data sets, it suffers from slow convergence on ill-conditioned problems, as can be seen for our model problem in Figure 1.

### 2.2.2 Stochastic Meta-Descent (SMD)

Schraudolph (1999, 2002) accelerates SGD by giving each system parameter its own gain:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{\eta}_t \cdot \nabla f(\boldsymbol{\theta}_t, \boldsymbol{X}_t), \qquad (8)$$

where $\cdot$ denotes Hadamard (element-wise) multiplication. The gain vector $\boldsymbol{\eta}_t$ is adapted by a dual gradient descent in the same objective, leading to the update

$$\boldsymbol{\eta}_t = \boldsymbol{\eta}_{t-1} \cdot \max[\tfrac{1}{2}, 1 - \mu\,\nabla f(\boldsymbol{\theta}_t, \boldsymbol{X}_t) \cdot \boldsymbol{v}_t], \qquad (9)$$

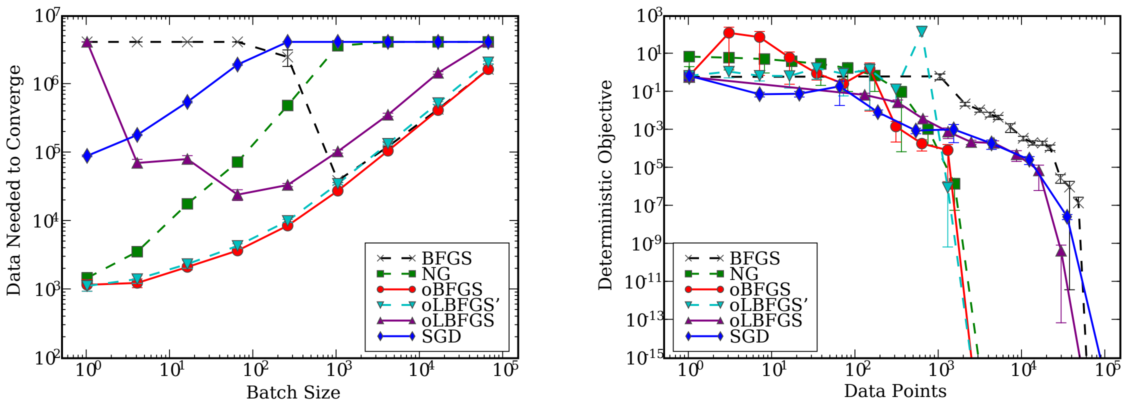



Figure 1: Average performance (with standard errors) of stochastic gradient methods on our model problem over 10 matched random replications. Left: number of data points needed to reach $f(\boldsymbol{\theta}) \leq 10^{-15}$, up to a limit of $2^{22}$, *vs.* batch size $b$. Right: deterministic loss $f(\boldsymbol{\theta})$ *vs.* number of data points seen, at the optimal batch size for each algorithm. All methods used $\eta_t = b/(b+2)$ (tuned for best performance of SGD) except NG, which required (7) with $\eta_0 = 1$ and $\tau = 100$. BFGS parameters: $\lambda = 0, c = 0.1; m = 4$ (oLBFGS) *resp.* 10 (oLBFGS').

where $\mu \geq 0$ is a scalar tuning parameter, and the auxiliary vector $\boldsymbol{v}$ is computed by the iterative update

$$\boldsymbol{v}_{t+1} = \lambda \boldsymbol{v}_t - \boldsymbol{\eta}_t \cdot [\nabla f(\boldsymbol{\theta}_t, \boldsymbol{X}_t) + \lambda \boldsymbol{H}_t \boldsymbol{v}_t], \qquad (10)$$

with $0 \leq \lambda \leq 1$ another scalar tuning parameter, and $\boldsymbol{H}_t$ the instantaneous Hessian at time $t$. Since $\boldsymbol{H}_t \boldsymbol{v}_t$ can be computed very efficiently (Schraudolph, 2002), SMD still takes only $O(n)$ space and time per iteration. It improves upon SGD by providing an adaptive gain decay, and handling some (but not all) forms of ill-conditioning. On our model problem, however, its performance essentially equaled that of SGD.

#### 2.2.3 Natural Gradient Descent (NG)

The natural gradient (NG) algorithm (Amari et al., 2000) incorporates the Riemannian metric tensor $\boldsymbol{G}_t := \mathbb{E}_{\boldsymbol{X}}[\nabla f(\boldsymbol{\theta}_t, \boldsymbol{X}_t) \nabla f(\boldsymbol{\theta}_t, \boldsymbol{X}_t)^\top]$ into the stochastic gradient update:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \hat{\boldsymbol{G}}_t^{-1} \nabla f(\boldsymbol{\theta}_t, \boldsymbol{X}_t), \qquad (11)$$

with gains $\eta_t$ typically set by (7), and $\hat{\boldsymbol{G}}_t$ an estimate of $\boldsymbol{G}_t$ updated via[1]

$$\hat{\boldsymbol{G}}_{t+1} = \frac{t-1}{t} \hat{\boldsymbol{G}}_t + \frac{1}{t} \nabla f(\boldsymbol{\theta}_t, \boldsymbol{X}_t) \nabla f(\boldsymbol{\theta}_t, \boldsymbol{X}_t)^\top. \qquad (12)$$

The Sherman-Morrison formula can be employed to directly update $\hat{\boldsymbol{G}}_t^{-1}$, reducing the computational cost of NG to $O(n^2)$ space and time per iteration — still prohibitively expensive for large $n$. Where it is affordable, NG greatly benefits from the incorporation of second-order information, as Figure 1 illustrates.

[1] Note that the use of a running average here is specifically optimized for our quadratic model, where $\boldsymbol{H} = $ const.

## 3 THE (L)BFGS ALGORITHM

We review BFGS in both full and memory-limited forms, and describe the changes needed for online use.

### 3.1 STANDARD BFGS METHOD

The BFGS algorithm (Nocedal and Wright, 1999) was developed independently by Broyden, Fletcher, Goldfarb, and Shanno. In the form we use here (Algorithm 1), it incrementally updates an estimate $\boldsymbol{B}_t$ of the inverse Hessian of the objective function. The rank-two update 3(h) minimizes a weighted Frobenius norm $\|\boldsymbol{B}_{t+1} - \boldsymbol{B}_t\|_W$ subject to the *secant equation* $\boldsymbol{s}_t = \boldsymbol{B}_{t+1} \boldsymbol{y}_t$, where $\boldsymbol{s}_t$ and $\boldsymbol{y}_t$ denote the most recent step along the optimization trajectory in parameter and gradient space, respectively.

$\boldsymbol{B}_t$ is then used to perform a quasi-Newton step 3(a), with gain $\eta_t$ determined by a line search 3(b). A line search obeying Wolfe conditions ensures that $(\forall t)\, \boldsymbol{s}_t^\top \boldsymbol{y}_t > 0$ and hence $(\forall t)\, \boldsymbol{B}_t \succ 0$. $\boldsymbol{B}_0$ is initialized to the identity but subsequently scaled by an estimate of the largest eigenvalue 3(f) of the inverse Hessian.

BFGS requires the same $O(n^2)$ space and time per iteration as NG but maintains a better model of loss curvature, which may permit a stochastic version of BFGS to converge faster than NG. However, extensive modifications are required to get BFGS to work online.

### 3.2 ONLINE BFGS METHOD

Algorithm 2 shows our online BFGS (oBFGS) method, with all modifications relative to standard BFGS (Algorithm 1) underlined. The changes required to get BFGS to work well with stochastic approximation fall

**Algorithm 1**: Standard BFGS Method

Given:

- objective $f$ and its gradient $\nabla f := \frac{\partial}{\partial \boldsymbol{\theta}} f(\boldsymbol{\theta})$;
- initial parameter vector $\boldsymbol{\theta}_0$;
- line search `linemin` obeying Wolfe conditions;
- convergence tolerance $\varepsilon > 0$;

1. $t := 0$;
2. $\boldsymbol{B}_0 = \boldsymbol{I}$;
3. while $\|\nabla f(\boldsymbol{\theta}_t)\| > \varepsilon$ :
   (a) $\boldsymbol{p}_t = -\boldsymbol{B}_t \nabla f(\boldsymbol{\theta}_t)$;
   (b) $\eta_t = \texttt{linemin}(f, \boldsymbol{\theta}_t, \boldsymbol{p}_t)$;
   (c) $\boldsymbol{s}_t = \eta_t \boldsymbol{p}_t$;
   (d) $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{s}_t$;
   (e) $\boldsymbol{y}_t = \nabla f(\boldsymbol{\theta}_{t+1}) - \nabla f(\boldsymbol{\theta}_t)$;
   (f) if $t = 0$ : $\boldsymbol{B}_t := \frac{\boldsymbol{s}_t^\top \boldsymbol{y}_t}{\boldsymbol{y}_t^\top \boldsymbol{y}_t} \boldsymbol{I}$;
   (g) $\varrho_t = (\boldsymbol{s}_t^\top \boldsymbol{y}_t)^{-1}$;
   (h) $\boldsymbol{B}_{t+1} = (\boldsymbol{I} - \varrho_t \boldsymbol{s}_t \boldsymbol{y}_t^\top) \boldsymbol{B}_t (\boldsymbol{I} - \varrho_t \boldsymbol{y}_t \boldsymbol{s}_t^\top) + \varrho_t \boldsymbol{s}_t \boldsymbol{s}_t^\top$;
   (i) $t := t + 1$;
4. return $\boldsymbol{\theta}_t$.

**Algorithm 2**: Online BFGS Method

Given:

- stochastic approximation of convex objective $f$ and its gradient $\nabla f$ over data sequence $\boldsymbol{X}_t$;
- initial parameter vector $\boldsymbol{\theta}_0$;
- sequence of step sizes $\eta_t > 0$;
- parameters $0 < c \leq 1, \lambda \geq 0, \epsilon > 0$;

1. $t := 0$;
2. $\boldsymbol{B}_0 = \epsilon \boldsymbol{I}$;
3. while not converged:
   (a) $\boldsymbol{p}_t = -\boldsymbol{B}_t \nabla f(\boldsymbol{\theta}_t, \boldsymbol{X}_t)$;
   (b) (no line search)
   (c) $\boldsymbol{s}_t = \frac{\eta_t}{c} \boldsymbol{p}_t$;
   (d) $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{s}_t$;
   (e) $\boldsymbol{y}_t = \nabla f(\boldsymbol{\theta}_{t+1}, \boldsymbol{X}_t) - \nabla f(\boldsymbol{\theta}_t, \boldsymbol{X}_t) + \lambda \boldsymbol{s}_t$;
   (f) if $t = 0$ : $\boldsymbol{B}_t := \frac{\boldsymbol{s}_t^\top \boldsymbol{y}_t}{\boldsymbol{y}_t^\top \boldsymbol{y}_t} \boldsymbol{I}$;
   (g) $\varrho_t = (\boldsymbol{s}_t^\top \boldsymbol{y}_t)^{-1}$;
   (h) $\boldsymbol{B}_{t+1} = (\boldsymbol{I} - \varrho_t \boldsymbol{s}_t \boldsymbol{y}_t^\top) \boldsymbol{B}_t (\boldsymbol{I} - \varrho_t \boldsymbol{y}_t \boldsymbol{s}_t^\top) + c \varrho_t \boldsymbol{s}_t \boldsymbol{s}_t^\top$;
   (i) $t := t + 1$;
4. return $\boldsymbol{\theta}_t$.

into three groups which we shall elaborate on in turn: making do without line search, modifying the update of $\boldsymbol{B}_t$, and taking consistent gradient measurements.

### 3.2.1 BFGS without Line Search

Line searches are highly problematic in a stochastic setting, since the global validity of the criteria they employ (such as the Wolfe conditions) cannot be established from local subsamples of the problem.

Unlike conjugate gradient methods, however, BFGS does not require an exact line search to correctly update its curvature estimate: we can actually replace the line search with a gain schedule such as (7) with no undue effect, provided we can ensure $\boldsymbol{B} \succ 0$ by other means. For now we do this by restricting our attention to convex optimization problems, for which $(\forall t)\, \boldsymbol{s}_t^\top \boldsymbol{y}_t \geq 0$ holds (no negative eigenvalues of $\boldsymbol{H}$). Very small eigenvalues ($\boldsymbol{s}_t^\top \boldsymbol{y}_t \approx 0$) are dealt with by modifying the BFGS update to estimate the inverse of $\boldsymbol{H} + \lambda \boldsymbol{I}$, where $\lambda \geq 0$ is a model-trust region parameter. This is achieved by simply adding $\lambda \boldsymbol{s}_t$ to $\boldsymbol{y}_t$ in step 3(e).

Finally, without line search we need to explicitly ensure that the first parameter update — before $\boldsymbol{B}_0$ has been appropriately scaled in step 3(f) — does not cause any problems. This is done by multiplying $\boldsymbol{B}_0$ in step 2 with a very small $\epsilon > 0$, so that the first parameter update is likewise small. The value of $\epsilon$ is application-dependent but non-critical; we typically use $\epsilon = 10^{-10}$.

### 3.2.2 Modified BFGS Update

We have found empirically that scaling down the last term of the update 3(h) by a factor $0 < c \leq 1$ substantially improves the performance of oBFGS for small batch sizes. We compensate for the resulting scaling of $\boldsymbol{B}_t$ by dividing the step size $\eta_t$ by $c$ in step 3(c). Scaling strategies for $\boldsymbol{B}_t$ are known from conventional BFGS (Brodlie, 1977). We anticipate being able to determine the optimal value for $c$ analytically; in the experiments reported here we simply used $c = 0.1$ throughout.

### 3.2.3 Consistent Gradient Measurements

We also need to account for the fact that in the stochastic setting our gradient measurements are noisy. This means that a simple convergence test like $\|\nabla f(\boldsymbol{\theta}_t)\| > \varepsilon$ in Algorithm 1 must be replaced by a more robust one, for instance checking whether the stochastic gradient has remained below a given threshold for the last $k$ iterations.

Finally, and most importantly, care must be taken in the computation of $\boldsymbol{y}_t$ in step 3(e). A naive translation of the "difference of last two gradients" into the stochastic setting would compute

$$\nabla f(\boldsymbol{\theta}_{t+1}, \boldsymbol{X}_{t+1}) - \nabla f(\boldsymbol{\theta}_t, \boldsymbol{X}_t)\,, \tag{13}$$

which would allow sampling noise to enter the BFGS update. Figure 1 ("BFGS" curves) shows the disastrous consequences: even in our simple quadratic model this causes divergence for $b < 10^3$.

Instead we must compute the difference $\boldsymbol{y}_t$ of gradients on the *same* data sample $\boldsymbol{X}_t$ used to compute the step $\boldsymbol{s}_t$. Although this doubles the number of gradient calculations, the extra computation is well spent: properly implemented, online BFGS ("oBFGS" in Figure 1) outperforms natural gradient for all batch sizes.

### 3.3 LIMITED-MEMORY BFGS

Limited-memory BFGS (LBFGS) is a variant of BFGS designed for solving large-scale optimization problems where the $O(n^2)$ cost of storing and updating $\boldsymbol{B}_t$ would be prohibitively expensive. In LBFGS the estimation of the Hessian inverse is based on only the last $m$ steps in parameter and gradient space; the quasi-Newton direction is obtained directly from these via a matrix-free approach (Algorithm 3). A conventional implementation of LBFGS would thus omit steps 2 and 3(f)-3(h) from Algorithm 1, maintain a ring buffer of the last $m$ vectors $\boldsymbol{s}$ and $\boldsymbol{y}$, and replace step 3(a) with Algorithm 3. This reduces computational cost to $O(mn)$ space and time per iteration.

It is straightforward to implement an LBFGS variant of our oBFGS algorithm: we simply modify Algorithm 2 analogous to the above, and replace step 3 of the LBFGS direction update (Algorithm 3) by

$$\boldsymbol{p}_t := \begin{cases} \epsilon\, \boldsymbol{p}_t & \text{if } t = 0; \\ \dfrac{\boldsymbol{p}_t}{\min(t,m)} \displaystyle\sum_{i=1}^{\min(t,m)} \frac{\boldsymbol{s}_{t-i}^\top \boldsymbol{y}_{t-i}}{\boldsymbol{y}_{t-i}^\top \boldsymbol{y}_{t-i}} & \text{otherwise.} \end{cases} \tag{14}$$

This ensures that the first parameter update is small (*cf.* step 2 of Algorithm 2), and improves online performance by averaging away some of the sampling noise.

Figure 1 shows that for $m = 4$ our online LBFGS algorithm (oLBFGS) performs well down to $b \approx 100$ but degrades for smaller batch sizes. This is not surprising considering that the curvature estimate is now based on only 4 noisy measurements of the objective. Fortunately the situation improves rapidly with increasing buffer size: for $m = 10$ (oLBFGS') performance is close to that of full online BFGS for all batch sizes.[2]

[2] Note that for $m > n$ LBFGS is computationally more expensive than full BFGS. For higher-dimensional problems, however, the beneficial effect of increasing $m$ will be realized well before approaching this anomalous regime.

**Algorithm 3**: LBFGS Direction Update

Given:

- integers $m > 0, t \geq 0$;
- $\forall\ i = 1, 2, \ldots, \min(t, m)$ : vectors $\boldsymbol{s}_{t-i}$ and $\boldsymbol{y}_{t-i}$ from Algorithm 1;
- current gradient $\nabla f(\boldsymbol{\theta}_t)$ of objective $f$;

1. $\boldsymbol{p}_t := -\nabla f(\boldsymbol{\theta}_t)$;
2. for $i := 1, 2, \ldots, \min(t, m)$ :
   (a) $\alpha_i = \dfrac{\boldsymbol{s}_{t-i}^\top \boldsymbol{p}_t}{\boldsymbol{s}_{t-i}^\top \boldsymbol{y}_{t-i}}$;
   (b) $\boldsymbol{p}_t := \boldsymbol{p}_t - \alpha_i \boldsymbol{y}_{t-i}$;
3. if $t > 0$ : $\boldsymbol{p}_t := \dfrac{\boldsymbol{s}_{t-1}^\top \boldsymbol{y}_{t-1}}{\boldsymbol{y}_{t-1}^\top \boldsymbol{y}_{t-1}} \boldsymbol{p}_t$;
4. for $i := \min(t, m), \ldots, 2, 1$ :
   (a) $\beta = \dfrac{\boldsymbol{y}_{t-i}^\top \boldsymbol{p}_t}{\boldsymbol{y}_{t-i}^\top \boldsymbol{s}_{t-i}}$;
   (b) $\boldsymbol{p}_t := \boldsymbol{p}_t + (\alpha_i - \beta)\boldsymbol{s}_{t-i}$;
5. return $\boldsymbol{p}_t$.

## 4 EXPERIMENTS

Having established the utility of our online BFGS algorithms on a simple quadratic stochastic model, we now turn to more challenging and realistic — albeit still convex — optimization problems.

### 4.1 NON-REALIZABLE QUADRATIC

Our quadratic objective (2) models *realizable* problems, *i.e.*, those where the loss at the optimum reaches zero for all inputs:

$$(\forall \boldsymbol{X})\ f(\boldsymbol{\theta}^*, \boldsymbol{X}) \ = \ 0 \tag{15}$$

Of greater practical relevance are non-realizable problems, in which the optimum carries a non-zero loss reflecting the best compromise between conflicting demands placed on the model by the data. Following Schraudolph and Graepel (2003) we model this by incorporating, along with each data sample $\boldsymbol{X}_t$, an i.i.d. Gaussian random vector $\boldsymbol{\nu}_t$ with zero mean and variance $\mathbb{E}_t[\boldsymbol{\nu}_t \boldsymbol{\nu}_t^\top] = \sigma^2 \boldsymbol{I}$ into our objective:

$$\begin{aligned} f(\boldsymbol{\theta}, \boldsymbol{X}_t) \ &= \ \frac{1}{2\,b}\, \boldsymbol{e}_t(\boldsymbol{\theta})^\top \boldsymbol{e}_t(\boldsymbol{\theta})\,, \quad \text{where} \\ \boldsymbol{e}_t(\boldsymbol{\theta}) \ &:= \ \boldsymbol{X}_t^\top \boldsymbol{J}^\top (\boldsymbol{\theta} - \boldsymbol{\theta}^*) \ + \ \boldsymbol{\nu}_t\,. \end{aligned} \tag{16}$$

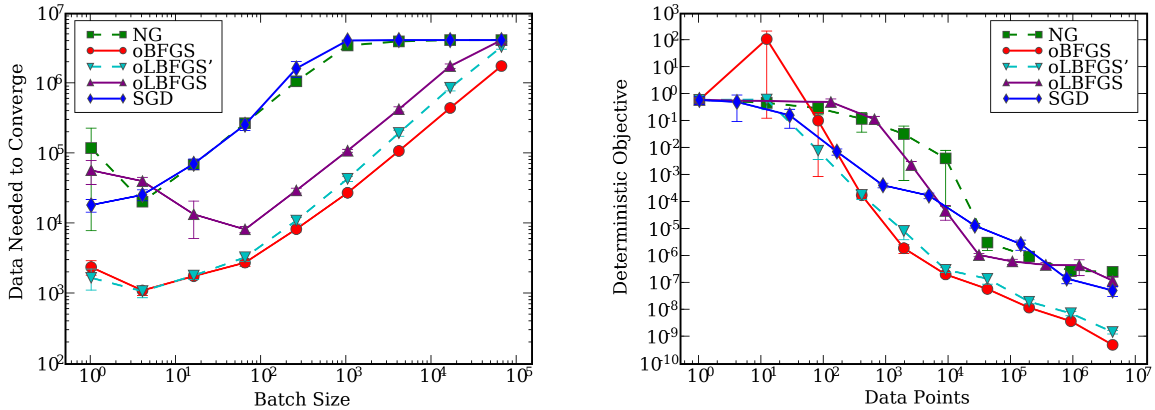



Figure 2: Average performance (with standard errors) of stochastic gradient methods on the non-realizable quadratic (16) with $\sigma = 10^{-2}$ over 10 matched random replications. Left: number of data points needed to converge to $f(\boldsymbol{\theta}) < 10^{-5}$, up to a limit of $2^{22}$, *vs.* batch size $b$. Right: deterministic loss $f(\boldsymbol{\theta})$ *vs.* number of data points seen, at the optimal batch size for each algorithm. All methods used (7) with $\eta_0$ and $\tau$ tuned for fastest convergence at small batch sizes: $\eta_0 = b/(b+2), \tau = 10^4$ (SGD); $\eta_0 = 0.04, \tau = 20$ (NG); $\eta_0 = b/(b+2), \tau = 20$ (oBFGS); $\eta_0 = 0.1 \cdot b/(b+2), \tau = 2 \cdot 10^4$ (oLBFGS); $\eta_0 = b/(b+2), \tau = 10$ (oLBFGS'); $\lambda, c, m$ as in Figure 1.

This makes the expected loss at the optimum $\boldsymbol{\theta}^*$ be

$$\mathbb{E}_t[f(\boldsymbol{\theta}^*, \boldsymbol{X}_t)] \;=\; \frac{1}{2\,b}\,\mathbb{E}_t[\boldsymbol{\nu}_t^\top \boldsymbol{\nu}_t] \;=\; \tfrac{1}{2}\,\sigma^2. \tag{17}$$

Moreover, the presence of $\boldsymbol{\nu}_t$ makes it impossible to determine $\boldsymbol{\theta}^*$ precisely from a finite data sample: the smaller the batch size $b$, the greater (for a given $\sigma$) the uncertainty in $\boldsymbol{\theta}^*$. Annealing the gains $\eta_t$ as in (7) addresses this by effectively averaging the gradient over progressively larger stretches of data.

Figure 2 shows our experimental results for optimizing this non-realizable objective. Because the noise term $\boldsymbol{\nu}_t$ inflates the metric tensor $\boldsymbol{G}_t$, natural gradient overestimates the curvature, and ends up performing no better than SGD here. BFGS, by contrast, bases its curvature estimate on *differences* of gradient measurements; as long as these are consistent (Section 3.2.3) any data-dependent noise or bias terms will thus be cancelled out.

Consequently, oBFGS greatly outperforms both SGD and NG here, converging about 20 times faster at the convenient mini-batch size of $b = 4$. The performance of oLBFGS with small buffer ($m = 4$) degrades for batch sizes below $b = 64$; a more generous buffer ($m = 10$), however, restores it to the level of full oBFGS.

## 4.2 CONDITIONAL RANDOM FIELDS

Conditional Random Fields (CRFs) have recently gained popularity in the machine learning community (Lafferty et al., 2001; Sha and Pereira, 2003; Kumar and Hebert, 2004). Conventional algorithms for batch CRF training — that is, penalized maximum likelihood parameter estimation — include generalized iterative scaling (GIS), conjugate gradient (CG), and limited-memory BFGS (Sha and Pereira, 2003). Vishwanathan et al. (2006) have recently shown that first-order stochastic gradient methods can greatly outperform the conventional batch algorithms.

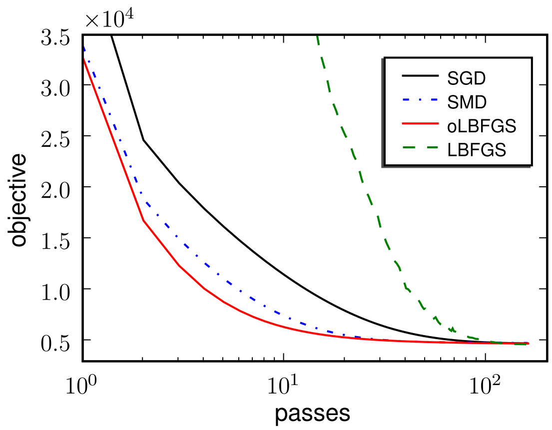
### 4.2.1 Experimental Tasks

We replicate two experiments by Vishwanathan et al. (2006) which apply 1-D chain CRFs to problems in natural language processing, using their software — an enhanced version of Taku Kudo's `CRF++` code[3] — and following their choice of CRF features, tuning parameter values, and optimization methods.

Due to the high-dimensional nature of these CRFs (over $10^5$ parameters), neither full BFGS nor natural gradient methods can be used here. Since the CRF parameter estimation problem is convex, however, we can apply our online LBFGS algorithm to it. To cope with regions of low curvature, we employ a model-trust region parameter $\lambda > 0$. The specific tasks were:

**CoNLL-2000 Base NP Chunking** (Sang and Buchholz, 2000): Text chunking, an intermediate step towards full parsing, divides a text into syntactically correlated chunks of words. The training set consists of 8936 sentences, each word annotated automatically with part-of-speech (POS) tags. The task is to label each word with a label indicating whether it lies outside, starts, or continues a chunk.

[3] Note that the line search used by `CRF++` for LBFGS does not guarantee a monotonic decrease in the objective.



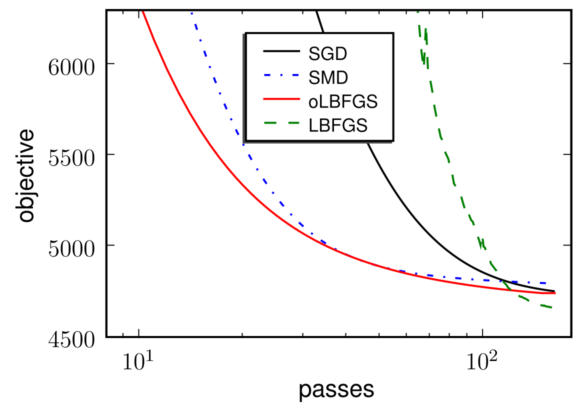




Figure 3: Left: loss on *vs.* number of passes through the CoNLL-2000 training set, for 4 gradient methods. Right: enlargement of final portion of the curves. Parameter values: SGD: $\eta = 0.1$; SMD: $\boldsymbol{\eta}_0 = 0.1, \mu = 0.1, \lambda = 1$; LBFGS: $m = 5$; oLBFGS: $m = 5$, (7) with $\tau = 10^4, \eta_0 = \lambda = 1$.

**BioNLP/NLPBA-2004 Shared Task** (Kim et al., 2004): This problem involves biomedical named-entity recognition on the GENIA corpus, aiming to identify and classify molecular biology terms in 18546 sentences of Medline abstracts.

Due to space constraints, we will present only the CoNLL-2000 experiment below; our results on the BioNLP/NLPBA-2004 task were analogous.

#### 4.2.2 Results on CoNLL-2000 Task

Figure 3 shows that oLBFGS initially tracks the performance of SGD but asymptotically achieves the lowest loss of the stochastic methods. Eventually (and not surprisingly) it is surpassed by the much slower but noise-free batch LBFGS algorithm. Whether the stochastic methods ultimately converge on the same solution as batch LBFGS is not clear at this point.

Figure 4 (left) shows that the generalization performance achieved by batch LBFGS after about 130 passes through the data — an F-score of 93.6% — is reached by oLBFGS in about 30 passes, and by SMD in just 7. An interesting observation here is that the algorithms differ substantially in their generalisation ability at a given loss level (Figure 4, right). It has been argued that stochastic approximation acts as a regularizer (Neuneier and Zimmermann, 1998, p. 397); our results illustrate how the utility of this effect depends on the particular stochastic gradient method used.

## 5 DISCUSSION

We have developed stochastic variants of the BFGS and LBFGS quasi-Newton methods, suitable for online optimization of convex functions. Experiments on realizable and non-realizable quadratic objectives show that our methods can greatly outperform other stochastic gradient algorithms, including a well-tuned natural gradient method. Unlike natural gradient, our oLBFGS algorithm scales well to very high-dimensional problems, such as parameter estimation for conditional random fields in natural language processing. One limitation is that for very sparse data, oLBFGS may require a substantial buffer size $m$ to produce a non-degenerate inverse curvature estimate.

We are now working to extend our approach to local optimization of non-convex objectives as well. We can already handle negative curvature by taking the absolute value of $\rho_t$ for updating $\boldsymbol{B}_t$ in Algorithm 2. A general method for nonlinear optimization, however, will also require online adaptation of gain $\eta_t$ and model-trust region parameter $\lambda$. We are looking into modifying stochastic gain adaptation methods such as SMD (Schraudolph, 1999, 2002) for this purpose.

We are also pursuing analytical proofs of convergence for our algorithms. Although such proofs are known for both first-order stochastic gradient and batch BFGS methods, extending them to oBFGS has proven challenging. Bottou and LeCun (2005) give general convergence conditions for second-order stochastic gradient methods; unfortunately they include that $\boldsymbol{B}_t \to \boldsymbol{H}^{-1}$ as $t \to \infty$, which does not hold for our algorithms. We are developing an alternative path to establish the convergence of oBFGS, based on the work of Robbins and Siegmund (1971).

This analysis should also provide insight into the free parameters ($\eta_t$, $c$, and $\lambda$) of the algorithm. Although online (L)BFGS does not require elaborate parameter tuning, we expect further improvements from developing ways to automatically set and adapt them.

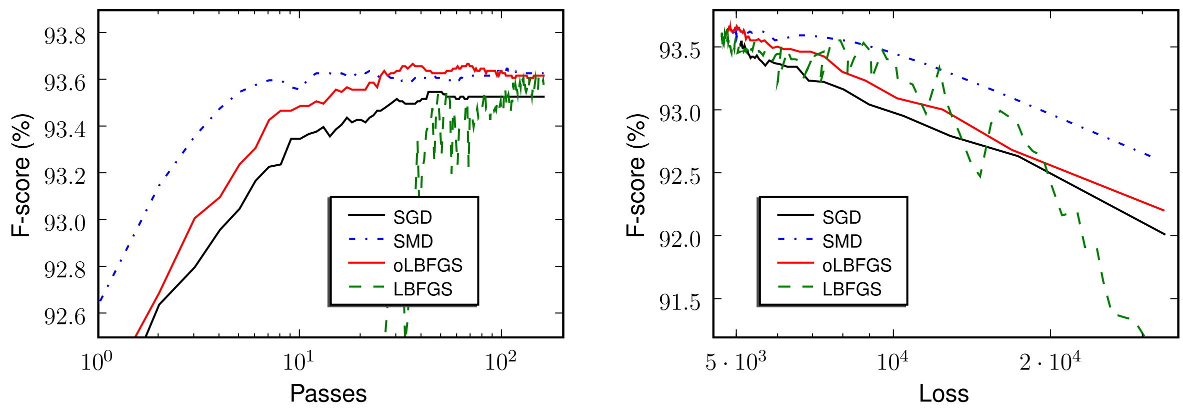



Figure 4: F-score on the CoNLL-2000 test set *vs.* number of passes through (left) *resp.* loss on (right) the training set, for the same experiment as in Figure 3.


## Acknowledgements

National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Center of Excellence program. This work is supported by the IST Program of the European Community, under the Pascal Network of Excellence, IST-2002-506778.


## References


S.-i. Amari, H. Park, and K. Fukumizu. Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural Computation*, 12(6):1399–1409, 2000.

L. Bottou and Y. LeCun. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21(2):137–151, 2005.

K. W. Brodlie. An assessment of two approaches to variable metric methods. *Mathematical Programming*, 12:344–355, 1977.

J.-D. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi, and N. Collier. Introduction to the bio-entity recognition task at JNLPBA. In *Proc. Intl. Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA)*, pages 70–75, Geneva, Switzerland, 2004.

S. Kumar and M. Hebert. Discriminative fields for modeling spatial dependencies in natural images. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, 2004.

J. D. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data. In *Proceedings of International Conference on Machine Learning*, volume 18, pages 282–289, San Francisco, CA, 2001. Morgan Kaufmann.

M. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.

R. Neuneier and H. G. Zimmermann. How to train neural networks. In G. B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, chapter 17, pages 373–423. Springer Verlag, Berlin, 1998.

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

G. V. Puskorius and L. A. Feldkamp. Decoupled extended Kalman filter training of feedforward layered networks. In *Proc. Intl. Joint Conf. on Neural Networks*, volume I, pages 771–777, Seattle, WA, 1991. IEEE.

H. E. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

H. E. Robbins and D. O. Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In *Proc. Sympos. Optimizing Methods in Statistics*, pages 233–257, Ohio State Univ., Columbus, Ohio, 1971. Academic Press, New York.

E. F. T. K. Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *Proc. Conf. Computational Natural Language Learning*, pages 127–132, Lisbon, Portugal, 2000.

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

N. N. Schraudolph. Local gain adaptation in stochastic gradient descent. In *Proc. Intl. Conf. Artificial Neural Networks*, pages 569–574, Edinburgh, Scotland, 1999. IEE, London.

N. N. Schraudolph and T. Graepel. Combining conjugate direction methods with stochastic approximation of gradients. In C. M. Bishop and B. J. Frey, editors, *Proc. 9th Intl. Workshop Artificial Intelligence and Statistics*, pages 7–13, Key West, Florida, 2003. ISBN 0-9727358-0-1.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL*, pages 213–220, Edmonton, Canada, 2003. Association for Computational Linguistics.

S. V. N. Vishwanathan, N. N. Schraudolph, M. Schmidt, and K. Murphy. Accelerated training conditional random fields with stochastic gradient methods. In *Proc. Intl. Conf. Machine Learning*, pages 969–976, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-383-2.