Stochastic Newton and quasi-Newton Methods for Large-Scale Convex and Non-Convex Optimization

Donald Goldfarb

Department of Industrial Engineering and Operations Research
Columbia University

Joint with Robert Gower, Peter Richtárik, Shiqian Ma, Xiao Wang and Wei Liu

U.S.-Mexico Workshop on Optimization and its Applications 2016
Merida, January 4, 2016

Outline

- Newton-like and quasi-Newton methods for convex stochastic optimization problems using limited memory block BFGS updates.
- Quasi-Newton methods for nonconvex stochastic optimization problems using damped limited memory BFGS updates.
- In both cases the objective functions can be expressed as the sum of a huge number of functions of an extremely large number of variables.
- We present numerical results on problems from machine learning.

Related work on L-BFGS for Stochastic Optimization

- P1 N.N. Schraudolph, J. Yu and S.Günter. A stochastic quasi-Newton method for online convex optim. Int'l. Conf. AI & Stat., 2007
- P2 A. Bordes, L. Bottou and P. Gallinari. SGD-QN: Careful quasi-Newton stochastic gradient descent. JMLR vol. 10, 2009
- P3 R.H. Byrd, S.L. Hansen, J. Nocedal, and Y. Singer. A stochastic quasi-Newton method for large-scale optim. arXiv1401.7020v2, 2014
- P4 A. Mokhtari and A. Ribeiro. RES: Regularized stochastic BFGS algorithm. IEEE Trans. Signal Process., no. 10, 2014.
- P5 A. Mokhtari and A. Ribeiro. Global convergence of online limited memory BFGS. to appear in J. Mach. Learn. Res., 2015.
- P6 P. Moritz, R. Nishihara, M.I. Jordan. A linearly-convergent stochastic L-BFGS Algorithm, 2015 arXiv:1508.02087v1
- P7 X. Wang, S. Ma, D. Goldfarb and W. Liu. Stochastic quasi-Newton methods for nonconvex stochastic optim. 2015, submitted.

(the first 6 papers are for strongly convex problems, the last one is for nonconvex problems)

Stochastic optimization

- Stochastic optimization

$$\min \; f(x) = \mathbb{E}[f(x,\xi)], \quad \xi \text{ is random variable}$$

- Or finite sum (with $f_i(x) \equiv f(x,\xi_i)$ for $i = 1,\ldots,n$ and very large n)

$$\min \; f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

- f and ∇f are very expensive to evaluate; e.g., SGD methods randomly choose a random subset $\mathcal{S} \subset [n]$ and evaluate

$$f_{\mathcal{S}}(x) = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} f_i(x) \quad \text{and} \quad \nabla f_{\mathcal{S}}(x) = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \nabla f_i(x)$$

- Essentially, only noisy info about f, ∇f and $\nabla^2 f$ is available
- Challenge: how to design a method that takes advantage of noisy 2nd-order information?

Part 1: Using 2nd-order information

- Assumption: $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$ is strongly convex and twice continuously differentiable.
- Choose (compute) a sketching matrix S_k (the columns of S_k are a set of directions).
- Following Byrd, Hansen, Nocedal and Singer, we do not use differences in noisy gradients to estimate curvature, but rather compute the action of the sub-sampled Hessian on S_k. i.e.,
- compute $Y_k = \frac{1}{|\mathcal{T}|}\sum_{i\in\mathcal{T}} \nabla^2 f_i(x) S_k$, where $\mathcal{T} \subset [n]$.
- We choose $\mathcal{T} = \mathcal{S}$

block BFGS

Given $H_k = B_k^{-1}$, the block BFGS method computes a "least change" update to the current approximation H_k to the inverse Hessian matrix $\nabla^2 f(x)$ at the current point x, by solving

$$\begin{aligned} \min \quad & \|H - H_k\| \\ \text{s.t.,} \quad & H = H^\top, \quad HY_k = S_k. \end{aligned}$$

This gives the updating formula (analgous to the updates derived by Broyden, Fletcher, Goldfarb and Shanno).

$$H_{k+1} = (I - S_k[S_k^\top Y_k]^{-1} Y_k^\top) H_k (I - Y_k [S_k^\top Y_k]^{-1} S_k^\top) + S_k [S_k^\top Y_k]^{-1} S_k^\top$$

or, by the Sherman-Morrison-Woodbury formula:

$$B_{k+1} = B_k - B_k S_k [S_k^\top B_k S_k]^{-1} S_k^\top B_k + Y_k [S_k^\top Y_k]^{-1} Y_k^\top$$

Limited Memory Block BFGS

After M block BFGS steps starting from H_{k+1-M}, one can express H_{k+1} as

$$
\begin{aligned}
H_{k+1} &= V_k H_k V_k^T + S_k \Lambda_k S_k^T \\
&= V_k V_{k-1} H_{k-1} V_{k-1}^T V_k + V_k S_{k-1} \Lambda_{k-1} S_{k-1}^T V_k^T + S_k \Lambda_k S_k^T \\
&\vdots \\
&= V_{k:k+1-M} H_{k+1-M} V_{k:k+1-M}^T + \sum_{i=k}^{k+1-M} V_{k:i+1} S_i \Lambda_i S_i^T V_{k:i+1}^T,
\end{aligned}
$$

where

$$V_k = (I - S_k \Lambda_k Y_k^T) \tag{1}$$

and $\Lambda_k = (S_k^T Y_k)^{-1}$ and $V_{k:i} = V_k \cdots V_i$.

Limited Memory Block BFGS

- Hence, when the number of variables d is large, instead of storing the $d \times d$ matrix H_k, we store the previous M block curvature pairs

$$\left(S_{k+1-M}, Y_{k+1-M}\right), \ldots, \left(S_k, Y_k\right).$$

- Then, analogously to the standard L-BFGS method, for any vector $v \in \mathbb{R}^d$, $H_k v$ can be computed efficiently using a two-loop block recursion (in $O(Mp(d+p)+p^3)$ operations), if all $S_i \in \mathbb{R}^{d \times p}$.

Intuition

- Limited memory - least change aspect of BFGS is important
- Each block update acts like a sketching procedure.

Choices for the Sketching Matrix S_k

We employ one of the following strategies

- Gaussian: $S_k \sim \mathcal{N}(0, I)$ has Gaussian entries sampled i.i.d at each iteration.
- Previous search directions s_i delayed: Store the previous L search directions $S_k = [s_{k+1-L}, \ldots, s_k]$ then update H_k only once every L iterations.
- Self-conditioning: Sample the columns of the Cholesky factors L_k of H_k (i.e., $L_k L_k^T = H_k$) uniformly at random. Fortunately we can maintain and update L_k efficiently with limited memory.

The matrix S is a sketching matrix, in the sense that we are sketching the, possibly very large equation $\nabla^2 f(x)H = I$ to which the solution is the inverse Hessian. Left multiplying by S^T compresses/sketches the equation yielding $S^T \nabla^2 f(x)H = S^T$.

Stochastic Variance Reduced Gradients

- Stochastic methods converge slowly near the optimum due to the variance of the gradient estimates $\nabla f_{\mathcal{S}}(x)$; hence requiring a decreasing step size.
- We use the control variates approach of Johnson and Zhang (2013) for a SGD method SVRG.
- It uses $\nabla f_{\mathcal{S}}(x_t) - \nabla f_{\mathcal{S}}(w_k) + \nabla f(w_k$, where w_k is a reference point, in place of $\nabla f_{\mathcal{S}}(x_t)$.
- w_k, and the full gradient, are computed after each full pass of the data, hence doubling the work of computing stochastic gradients.
- Other SGD variance reduction techniques have been recently proposes including the methods: SAG, SAGA, SDCA, S2GD.

The Basic Algorithm

Algorithm 0.1: Stochastic Variable Metric Learning with SVRG

Input: $H_{-1} \in \mathbb{R}^{d \times d}$, $w_0 \in \mathbb{R}^d$, $\eta \in \mathbb{R}_+$, s = subsample size, q = sample action size and m

for $k = 0, \ldots, max\_iter$ **do**
  $\mu = \nabla f(w_k)$
  $x_0 = w_k$
  for $t = 0, \ldots, m-1$ **do**
    Sample $\mathcal{S}_t, \mathcal{T}_t \subseteq [n]$ i.i.d from a distribution $\mathcal{S}$
    Compute the sketching matrix $S_t \in \mathbb{R}^{d \times q}$
    Compute $\nabla^2 f_{\mathcal{S}}(x_t) S_t$
    $H_t = \text{update\_metric}(H_{t-1}, S_t, \nabla^2 f_{\mathcal{T}}(x_t) S_t)$
    $d_t = -H_t \left(\nabla f_{\mathcal{S}}(x_t) - \nabla f_{\mathcal{S}}(w_k) + \mu\right)$
    $x_{t+1} = x_t + \eta d_t$
  end
  Option I: $w_{k+1} = x_m$
  Option II: $w_{k+1} = x_i$, i selected uniformly at random from $[m]$;
end

Convergence - Assumptions

There exist constants $\lambda, \Lambda \in \mathbb{R}_+$ such that

- f is λ–strongly convex

$$f(w) \geq f(x) + \nabla f(x)^T(w-x) + \frac{\lambda}{2}\|w-x\|_2^2, \qquad (2)$$

- f is Λ–smooth

$$f(w) \leq f(x) + \nabla f(x)^T(w-x) + \frac{\Lambda}{2}\|w-x\|_2^2, \qquad (3)$$

- These assumptions imply that

$$\lambda I \preceq \nabla^2 f_{\mathcal{S}}(w) \preceq \Lambda I, \quad \text{for all } x \in \mathbb{R}^d, \mathcal{S} \subseteq [n], \qquad (4)$$

- from which we can prove that there exist constants $\gamma, \Gamma \in \mathbb{R}_+$ such that for all k we have

$$\gamma I \preceq H_k \preceq \Gamma I. \qquad (5)$$

Linear Convergence

Theorem

Suppose that the Assumptions hold. Let w_* be the unique minimizer of $f(w)$. Then in our Algorithm, we have for all $k \geq 0$ that

$$\mathbb{E}f(w_k) - f(w_*) \leq \rho^k \mathbb{E}f(w_0) - f(w_*),$$

where the convergence rate is given by

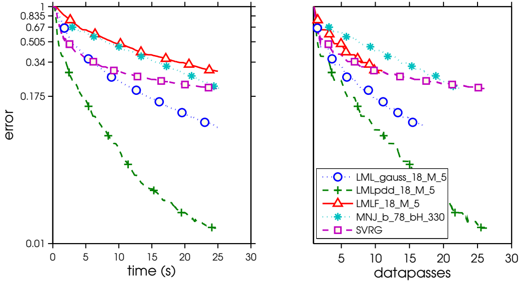
$$\rho = \frac{1/2m\eta + \eta\Gamma^2\Lambda(\Lambda - \lambda)}{\gamma\lambda - \eta\Gamma^2\Lambda^2} < 1,$$

assuming we have chosen $\eta < \gamma\lambda/(2\Gamma^2\Lambda^2)$ and that we choose m large enough to satisfy

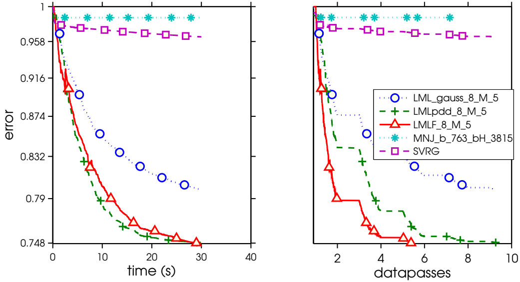
$$m \geq \frac{1}{2\eta\left(\gamma\lambda - \eta\Gamma^2\Lambda(2\Lambda - \lambda)\right)},$$

which is a positive lower bound given our restriction on η.

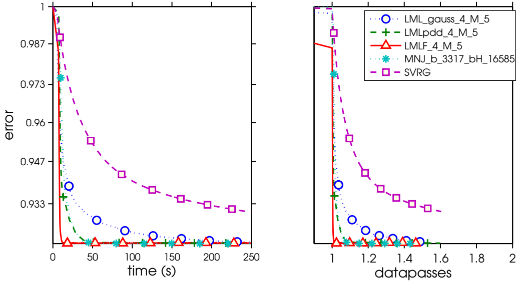
gisette-scale $d = 5,000$, $n = 6,000$



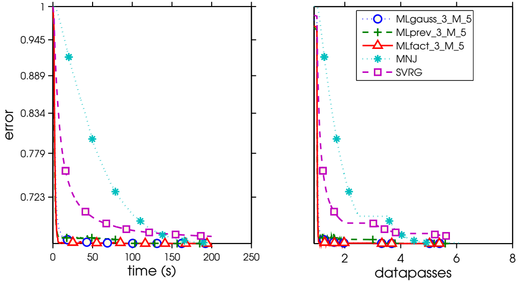
covtype-libsvm-binary $d = 54, n = 581,012$



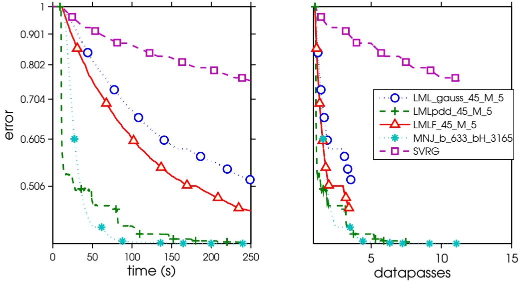
Higgs $d = 28, n = 11,000,000$



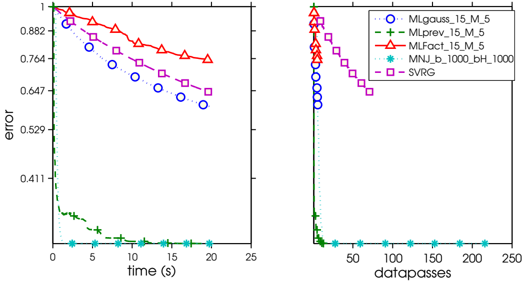
SUSY $d = 18, n = 3,548,466$



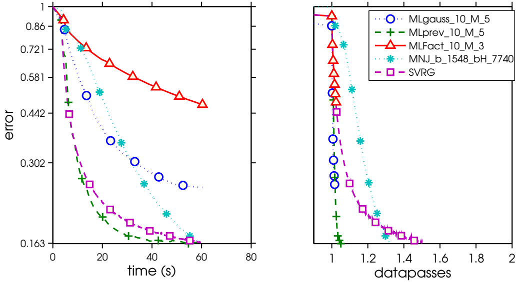
epsilon-normalized $d = 2,000$, $n = 400,000$



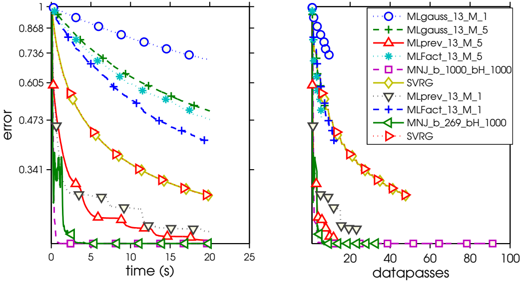
rcv1-training $d = 47,236$, $n = 20,242$



url-combined $d = 3,231,961, n = 2,396,130$



zero-real-sim-L2 $d = 20,958$, $n = 72,309$



Contributions

- *New metric learning framework.* A block BFGS framework for gradually learning the metric of the underlying function using a sketched form of the subsampled Hessian matrix
- *New limited memory block BFGS method.* May also be of interest for non-stochastic optimization
- *Several sketching matrix possibilities.*

Part 2: Nonconvex stochastic optimization

- Most stochastic quasi-Newton optimization methods are for strongly convex problems; this is needed to ensure a curvature condition required for the positive definiteness of B_k (H_k)
- This is not possible for nonconvex problem
- In deterministic setting, one can do line search to guarantee the curvature condition, and hence the positive definiteness of B_k (H_k)
- Line search is not possible for stochastic optimization
- To address these issues we develop a stochastic damped L-BFGS method:

Stochastic quasi-Newton (SQN) for nonconvex problem

$$\min f(x) \equiv \mathbb{E}[F(x,\xi)]$$

Assumptions

[AS1] f is continuously differentiable; f is bounded below; ∇f is Lipschitz continuous with constant L

[AS2] For any iteration k, we have stochastic gradient satisfies

$$\mathbb{E}_{\xi_k}[\nabla f(x_k,\xi_k)] = \nabla f(x_k)$$
$$\mathbb{E}_{\xi_k}[\|\nabla f(x_k,\xi_k) - \nabla f(x_k)\|^2] \leq \sigma^2$$

[AS3] Exist positive constants C_l, C_u, such that

$$C_l I \preceq H_k \preceq C_u I, \quad \text{for any } k$$

[AS4] H_k depends only on $\xi_{[k-1]}$, i.e., on all the random samples in iterations $1, 2, \ldots, k-1$.

How to generate H_k to satisfy AS3 and AS4?

- Let $y_k = \frac{1}{m}\sum_{i=1}^{m}(\nabla f(x_{k+1}, \xi_{k,i}) - \nabla f(x_k, \xi_{k,i}))$ and define

$$\bar{y}_k = \theta_k y_k + (1-\theta_k) B_k s_k,$$

where θ_k is calculated through:

$$\theta_k = \begin{cases} 1, & \text{if } s_k^\top y_k \geq 0.25 s_k^\top B_k s_k, \\ (0.75 s_k^\top B_k s_k)/(s_k^\top B_k s_k - s_k^\top y_k), & \text{if } s_k^\top y_k < 0.25 s_k^\top B_k s_k. \end{cases}$$

- Update H_k: (replace y_k by $\bar{y}_k$)

$$H_{k+1} = (I - \rho_k s_k \bar{y}_k^\top) H_k (I - \rho_k \bar{y}_k s_k^\top) + \rho_k s_k s_k^\top$$

where $\rho_k = 1/s_k^\top \bar{y}_k$

- Implement in a limited memory version

Numerical Experiments

- A nonconvex SVM problem with a sigmoid loss function

$$\min_{x \in \mathbb{R}^n} \quad f(x) := \mathbb{E}_{u,v}[1 - \tanh(v\langle x, u\rangle)] + \lambda\|x\|_2^2,$$

- $u \in \mathbb{R}^n$: feature vector; $v \in \{-1, 1\}$: corresponding label.
- $\lambda = 10^{-4}$ in our experiment
- RCV1 dataset: Reuters newswire articles from 1996-1997.
- A simplified version: 9625 articles classified into four categories "C15", "ECAT", "GCAT" and "MCAT", each with 2022, 2064, 2901 and 2638 articles, respectively.
- Binary classification: predict if an article is in "MCAT" and "ECAT".
- Label: 1 if a given word in "MCAT" or "ECAT", -1 otherwise.
- 60% of the articles - training data; 40% - testing data.
- Problem dimension: 29992 (number of distinct words)

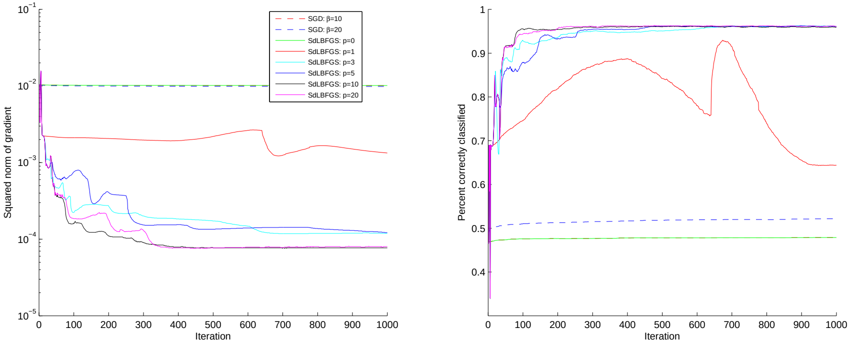


Figure: Comparison of SdLBFGS variants with different memory size on RCV1 dataset. The step size of SdLBFGS is $\alpha_k = 10/k$ and the batch size is $m = 100$.

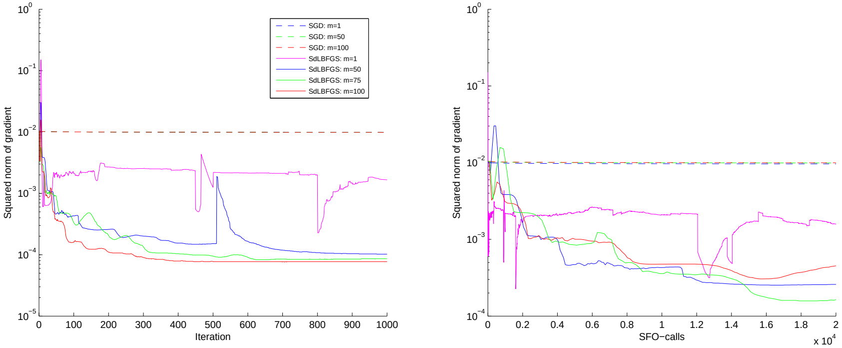


Figure: Comparison of SGD and SdLBFGS with different batch size on RCV1 dataset. For SdLBFGS the step size is $\alpha_k = 10/k$ and the memory size is $p = 10$. For SGD the step size is $\alpha_k = 20/k$.

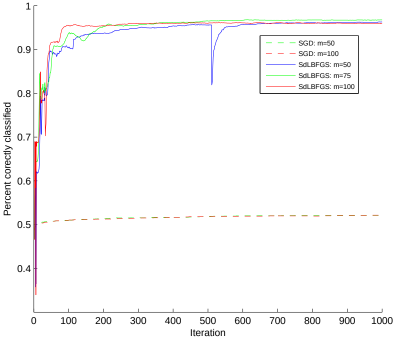


Figure: Comparison of correct classification percentage by SGD and SdLBFGS with different batch size on RCV1 dataset. For SdLBFGS the step size is $\alpha_k = 10/k$ and the memory size is $p = 10$. For SGD the step size is $\alpha_k = 20/k$.

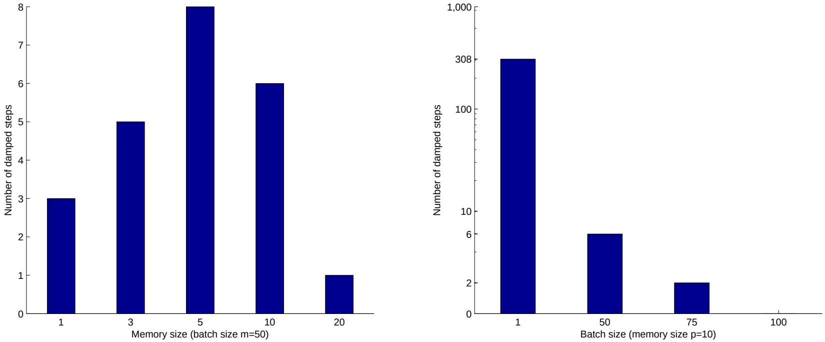


Figure: The average number of damped steps over 10 runs of SdLBFGS. Here the maximum number of iterations is set as 1000 and step size is $10/k$.

Contributions

- Our contributions:
  - A general framework of SQN for nonconvex problem
  - Convergence guarantee
  - Complexity analysis for random output and constant step size
  - Stochastic damped L-BFGS falls into the framework
- Future work for nonconvex problems:
  - develop a damped limited memory block BFGS method
  - Variance reduction techniques?