# Sampled Quasi-Newton Methods for Deep Learning

Albert S. Berahas, Majid Jahani, Martin Takáč
Lehigh University, Bethlehem, PA

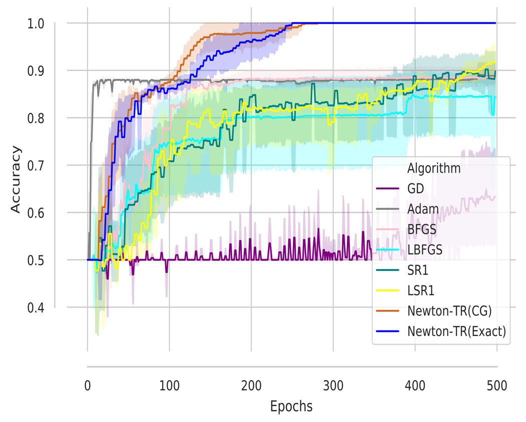
## Abstract

✓ Proposed two novel quasi-Newton methods that use sampling to construct Hessian approximations
✓ Proved theoretical guarantees of the proposed methods
✓ Showed the practical performance of the methods on deep learning tasks
✓ Discussed the implementation costs of the sampled quasi-Newton methods and compare them to the classical variants

## Introduction

$$\min_{w\in\mathbb{R}^d} F(w) := \frac{1}{n}\sum_{i=1}^{n} f(w;x^i,y^i) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

- $n$ and $d$ are large, and $F(.)$ is nonconvex
- First-order methods converge very slowly, and sometimes even fail to achieve 100% accuracy
- Methods that use the true Hessian are always able to achieve 100% in a few iterations; however, they are expensive
- The curvature information captured by classical quasi-Newton may not be adequate or useful
- **Our idea:** ***forget*** past curvature information and ***sample*** new curvature pairs at every iteration




## Literature Review

- **BFGS/LBFGS** : Broyden, 1967; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970, Nocedal, 1980; Liu & Nocedal, 1989; Gao and Goldfarb, 2018 Liu & Nocedal, 1989
- **SR1/LSR1** : Conn et al., 1991; Khalfan et al., 1993; Byrd et al., 1996; Lu, 1996; Brust et al., 2017;
- **Stochastic QN** : Schraudolph et al., 2007; Mokhtari & Ribeiro, 2015; Byrd et al., 2016; Berahas et al., 2016; Curtis, 2016; Gower et al. 2016;

## Quasi-Newton Methods

### BFGS and LBFGS

$$w_{k+1} = w_k - \alpha_k H_k \nabla F(w_k),$$

where $H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T$
and $\rho_k = \frac{1}{y_k^T s_k}$, $V_k = I - \rho_k y_k s_k^T$
and the curvature pairs $(s_k, y_k)$ :

$$s_k = w_k - w_{k-1},$$
$$y_k = \nabla F(w_k) - \nabla F(w_{k-1})$$

BFGS condition:

$$s^T y \geq \epsilon \|s\|^2 \qquad (1)$$

### SR1 and LSR1

$$w_{k+1} = w_k + p_k,$$

where $p_k$ is the minimizer of the following sub-problem

$\min_p m_k(p) = F(w_k) + \nabla F(w_k)^T p + \frac{1}{2} p^T B_k p,$
s.t. $\|p\| \leq \Delta_k,$

$\Delta_k$ is the trust region and $B_k$ is the SR1 Hessian approximation computed as

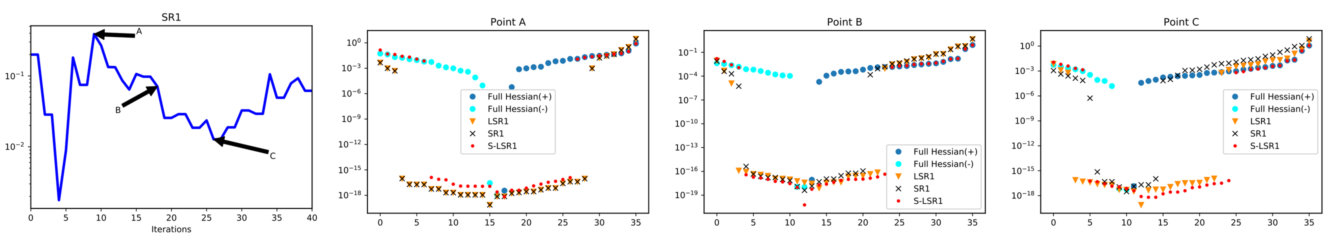
$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}.$$

SR1 condition:

$$|s^T(y - Bs)| \geq \epsilon \|s\| \|y - Bs\| \qquad (2)$$

## Sampled Quasi-Newton Methods

### Can one capture better curvature via sampling?




Comparison of the eigenvalues of SR1, LSR1 and S-LSR1 at points A, B and C for a toy classification problem.

**Algorithm 1** Compute new $(S, Y)$ curvature pairs

**Input:** $w$ (iterate), $m$ (memory), $r$ (sampling radius), $S = [\,]$, $Y = [\,]$
Compute $\nabla F(w)$
**for** $i = 1, 2, ..., m$ **do**
  Sample a random direction of unit length $\sigma_i$
  Sample point $\bar{w} = w + r\sigma_i$
  Set $s = w - \bar{w}$ and
  $y = \begin{cases} \nabla F(w) - \nabla F(\bar{w}), & \text{Option I} \\ \nabla^2 F(w) s, & \text{Option II} \end{cases}$
  Set $S = [S\ s]$ and $Y = [Y\ y]$
**end for**
**Output:** $S, Y$

- Sample points around the current iterate along random directions $\sigma_i$
- Option I requires $m$ gradient evaluations
- Option I is significantly more sensitive to the choice of the sampling radius
- Option II is **scale invariant** and needs a **single Hessian matrix product**
- Option II, $y$ curvature pairs can be calculated **simultaneously** and **efficiently on a GPU**

## Sampled LBFGS & Sampled LSR1

**Algorithm 2** Sampled LBFGS (S-LBFGS)

**Input:** $w_0$ (initial iterate), $m$ (memory), $r$ (sampling radius).
**for** $k = 0, 1, 2, ...$ **do**
  Compute new $(S_k, Y_k)$ pairs via Algorithm 1
  Compute $p_k = -H_k \nabla F(w_k)$
  Choose the steplength $\alpha_k > 0$
  Set $w_{k+1} = w_k + \alpha_k p_k$
**end for**

**Algorithm 3** Sampled LSR1 (S-LSR1)

**Input:** $w_0$ (initial iterate), $\Delta_0$ (initial trust region radius), $m$ (memory), $r$ (sampling radius).
**for** $k = 0, 1, 2, ...$ **do**
  Compute new $(S_k, Y_k)$ pairs via Algorithm 1
  Compute $B_{k+1}$
  Compute $p_k$ by solving the TR subproblem
  Compute $\rho_k = \frac{F(w_k) - F(w_k + p_k)}{m_k(0) - m_k(p_k)}$
  **if** $\rho_k \geq \eta_1$ **then**
    Set $w_{k+1} = w_k + p_k$
  **else**
    Set $w_{k+1} = w_k$
  **end if**
  $\Delta_{k+1} =$ `Adjust trust-region radius`$(\Delta_k, \rho_k)$
**end for**

Key diffrentiating elements with classical variants: (1) the way in which curvature pairs are created; (2) the location in the algorithm where the curvature pairs are constructed (i.e., even the first step is quasi-Newton)

## Convergence Analysis

### Sampled LBFGS - Strongly Convex Functions

**Assumption 1.** *$F$ is twice continuously differentiable.*

**Assumption 2.** *There exist positive constants $\mu$ and $L$ such that $\mu I \preceq \nabla^2 F(w) \preceq LI$, for all $w \in \mathbb{R}^d$.*

**Lemma 3.** *If Assumptions 1 and 2 hold, there exist constants $0 < \mu_1 \leq \mu_2$ such that the inverse Hessian approximations $\{H_k\}$ generated by Algorithm 2 satisfy,*

$$\mu_1 I \preceq H_k \preceq \mu_2 I, \qquad \text{for } k = 0, 1, 2, \ldots.$$

**Theorem 4.** *Suppose that Assumptions 1 and 2 hold, and let $F^\star = F(w^\star)$, where $w^\star$ is the minimizer of $F$. Let $\{w_k\}$ be the iterates generated by Algorithm 2, where $0 < \alpha_k = \alpha \leq \frac{\mu_1}{\mu_2^2 L}$, and $w_0$ is the starting point. Then for all $k \geq 0$,*

$$F(w_k) - F^\star \leq (1 - \alpha\mu\mu_1)^k [F(w_0) - F^\star].$$

### Sampled LBFGS - Nonconvex Functions

**Assumption 5.** *The function $F(w)$ is bounded below by a scalar $\widehat{F}$.*

**Assumption 6.** *The gradients of $F$ are $L$-Lipschitz continuous for all $w \in \mathbb{R}^d$.*

**Lemma 7.** *Suppose that Assumptions 1 and 6 hold. Let $\{H_k\}$ be the inverse Hessian approximations generated by Algorithm 2, with the modification that the inverse approximation update is performed using only curvature pairs that satisfy (1), for some $\epsilon > 0$, and $H_k = I$ if no curvature pairs satisfy (1). Then, there exist constants $0 < \mu_1 \leq \mu_2$ such that*

$$\mu_1 I \preceq H_k \preceq \mu_2 I, \qquad \text{for } k = 0, 1, 2, \ldots.$$

**Theorem 8.** *Suppose that Assumptions 1, 5 and 6 hold. Let $\{w_k\}$ be the iterates generated by Algorithm 2, with the modification that the inverse Hessian approximation update is performed using only curvature pairs that satisfy (1), for some $\epsilon > 0$, and $H_k = I$ if no curvature pairs satisfy (1), where $0 < \alpha_k = \alpha \leq \frac{\mu_1}{\mu_2^2 L}$, and $w_0$ is the starting point. Then, $\lim_{k\to\infty} \|\nabla F(w_k)\| = 0$, and, moreover, for any $\tau > 1$,*

$$\frac{1}{\tau}\sum_{k=0}^{\tau-1} \|\nabla F(w_k)\|^2 \leq \frac{2[F(w_0) - \widehat{F}]}{\alpha\mu_1\tau} \xrightarrow{\tau\to\infty} 0.$$

### Sampled LSR1

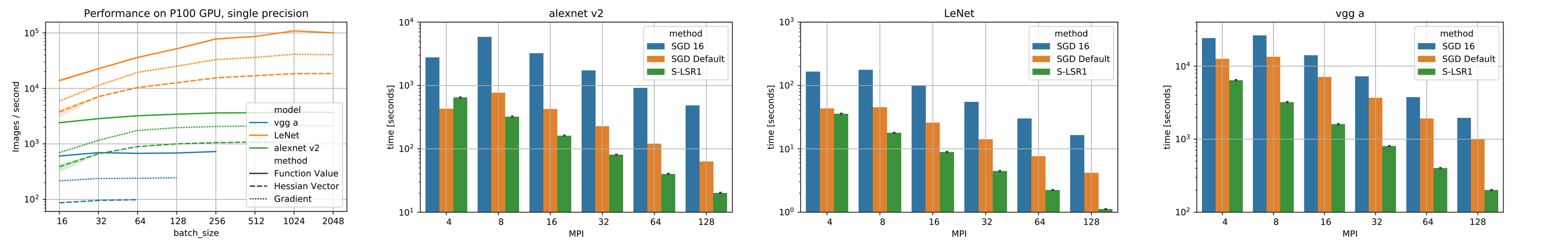
**Assumption 9.** *For all $k$, $m_k(0) - m_k(p_k) \geq \xi \|\nabla F(w_k)\| \min\left[\frac{\|\nabla F(w_k)\|}{\beta_k}, \Delta_k\right]$, where $\xi \in (0,1)$ and $\beta_k = 1 + \|B_k\|$.*

**Lemma 10.** *Suppose that Assumptions 1, 6 and 9 hold. Let $\{B_k\}$ be the Hessian approximations generated by Algorithm 3, with the modification that the approximation update is performed using only curvature pairs that satisfy (2), for some $\epsilon > 0$, and $B_k = I$ if no curvature pairs satisfy (2). Then, there exists a constant $\nu_2 > 0$ such that*

$$\|B_k\| \leq \nu_2, \qquad \text{for } k = 0, 1, 2, \ldots.$$

**Theorem 11.** *Suppose that Assumptions 1, 5, 6 and 9 hold. Let $\{w_k\}$ be the iterates generated by Algorithm 3, with the modification that the Hessian approximation update is performed using only curvature pairs that satisfy (2), for some $\epsilon > 0$, and $B_k = I$ if no curvature pairs satisfy (2). Then, $\lim_{k\to\infty} \|\nabla F(w_k)\| = 0$.*

## Distributed Computing




Performance (Images/second) as a function of batch size for different DNN models and operations on a single P100 GPU (left). Time (seconds) to complete 1 epoch of SG and to perform 1 iteration of S-LSR1 on a dataset with 1M images using varying number of MPI processes.

## Comparison of Computational Cost and Storage

- Number of line search iterations and CG iterations are denoted as $\kappa_{ls}$ and $\kappa_{tr}$, respectively

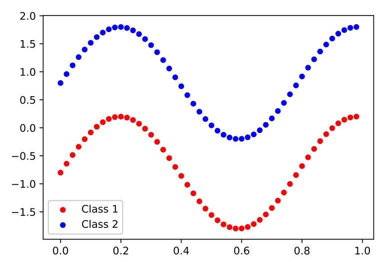
| method | computational cost | storage |
|---|---|---|
| BFGS | $nd + d^2 + \kappa_{ls} nd$ | $d^2$ |
| LBFGS | $nd + 4md + \kappa_{ls} nd$ | $2md$ |
| S-LBFGS | $nd + mnd + 4md + \kappa_{ls} nd$ | - |
| SR1 | $nd + d^2 + nd + \kappa_{tr} d^2$ | $d^2$ |
| LSR1 | $nd + nd + \kappa_{tr} md$ | $2md$ |
| S-LSR1 | $nd + mnd + nd + \kappa_{tr} md$ | - |

- The sampled quasi-Newton methods have **NO** storage requirements
- The per iteration cost of the sampled quasi-Newton methods is **COMPARABLE** to that of the classical limited memory variants
- In $m \ll n, d$ regime, the computational cost of the methods is $\mathcal{O}(nd)$

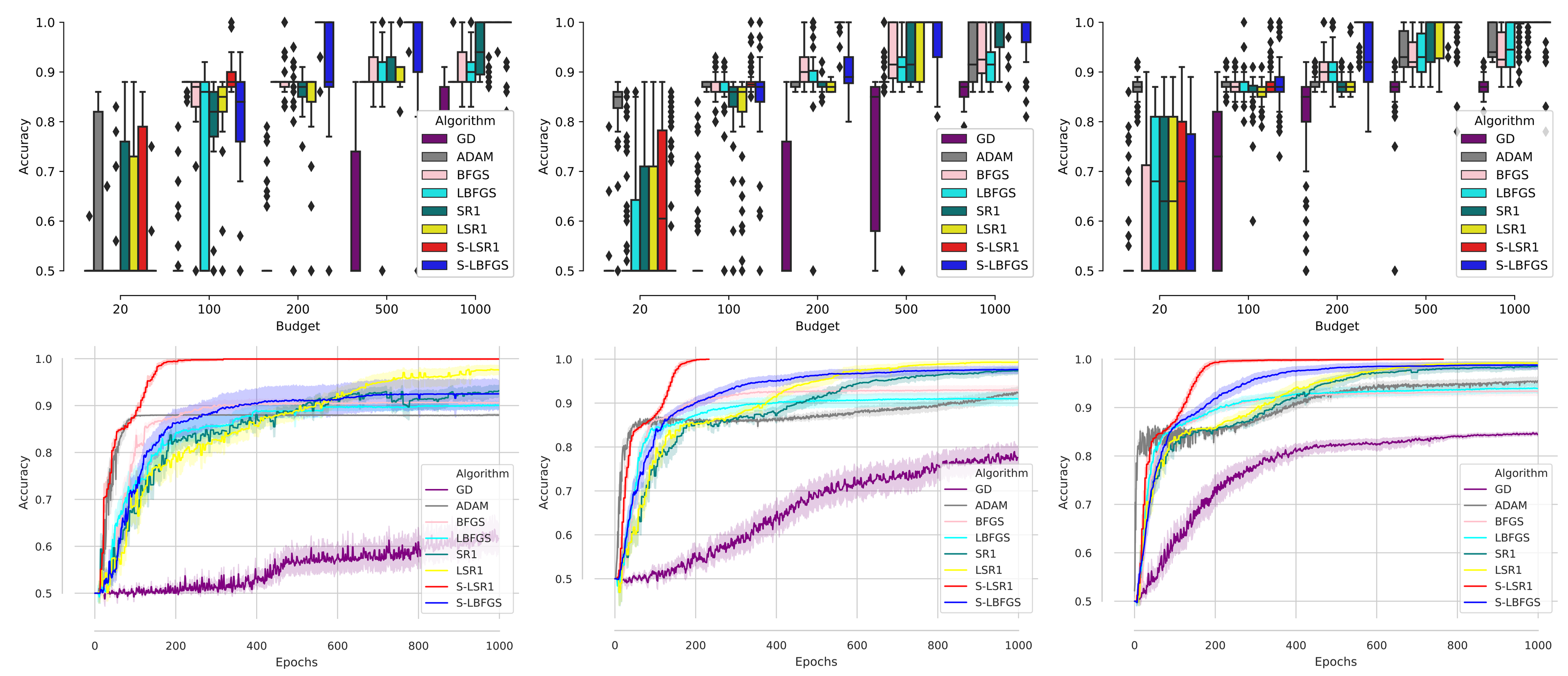
## Numerical Results

### Toy Classification Problem

- Two classes each with 50 data points
- Trained three FCNNs – `small`, `medium` and `large` – with sigmoid activation functions and 4 hidden layers

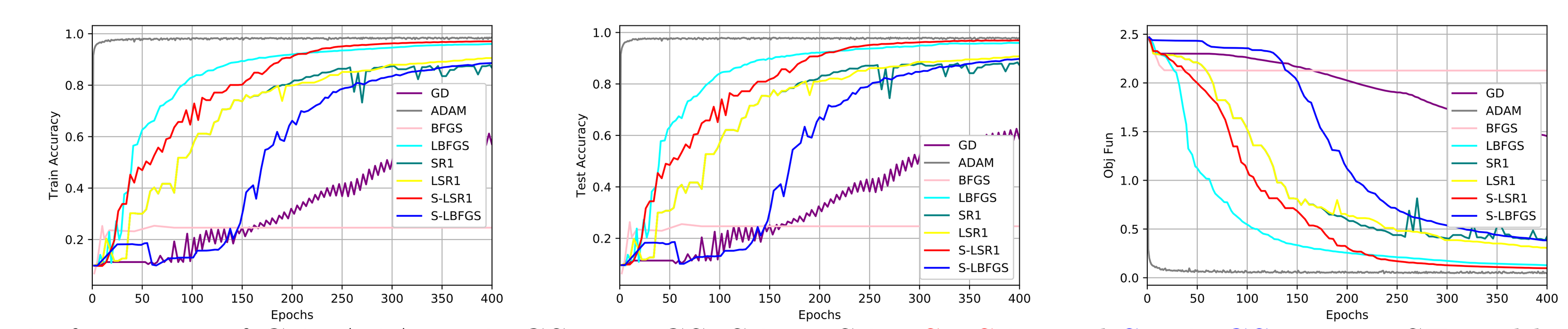



| network | structure | $d$ |
|---|---|---|
| `small` | 2-2-2-2-2-2 | 36 |
| `medium` | 2-4-8-8-4-2 | 176 |
| `large` | 2-10-20-20-10-2 | 908 |




Performance of GD, ADAM, BFGS, LBFGS, SR1, LSR1, S-LSR1 and S-LBFGS on toy classification problems. Networks: small (left); medium (middle); large (right).

### MNIST




Performance of GD, ADAM, BFGS, LBFGS, SR1, LSR1, S-LSR1 and S-LBFGS on MNIST problems.

## Future Work

- Extend our proposed methods to the stochastic setting (inexact gradients and/or Hessians)
- Extend and modify our methods to incorporate adaptive batch-sizes and memory
- Conduct a large scale numerical investigation of the proposed methods

## References

- Berahas, A. S., Jahani, M., & Takáč , M. (2019). Quasi-Newton Methods for Deep Learning. OPT2019.
- Jahani, M., Nazari, M., rusakov, S., Berahas, A. S., & Takáč , M. (2019). Scaling Up Quasi-Newton Algorithms: Communication Efficient Distributed SR1. arXiv preprint arXiv:1905.13096v1.
- Berahas, A. S., Jahani, M., & Takáč , M. (2019). Quasi-Newton Methods for Deep Learning: Forget the Past, Just Sample. arXiv preprint arXiv:1901.09997.
- Code: `https://github.com/OptMLGroup/SQN`

maj316@lehigh.edu