Stop Wasting My Gradients: Practical SVRG

Reza Babanezhad (UBC), Mohamed Osama Ahmed (UBC), Alim Virani (UBC), Mark Schmidt (UBC), Jakub Konečný (University of Edinburgh), Scott Sallinen (UBC)

Motivation and Overview of Contribution

- Huge proportion of ML model fitting problem involve minimizing finite sum:

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x).$$

  - Least square, logistic regression, conditional random fields, deep neural network, etc.
- Classic *full gradient* (FG) and *stochastic gradient* (SG) methods:
  - Have to choose between fast convergence (FG) and cheap iterations (SG).
- *Stochastic average gradient* (SAG):
  - Fast convergence *and* cheap iterations, but high memory requirement.
- *Stochastic variance-reduced gradient* (SVRG):
  - Fast convergence *and* cheap iterations, no memory requirement but many more gradients than SAG.
  - $2m+n$ gradient evaluations in SVRG for every m gradient evaluations in SAG.
- Our contributions:
  - Convergence of SVRG with noisy gradients.
  - Reducing gradient evaluations using batches.
  - Reducing gradient evaluations using support vectors.
  - Analysis of regularized SVRG iteration.
  - Alternative mini-batch strategies.
  - Generalization error of the method.

SVRG Algorithm and Convergence Rate

- Assumptions:
  - f is μ-strongly convex.
  - f_i is convex and f_i' is Lipschitz-continuous with constant L.
- SVRG ‘inner’ update is m variance-reduced SG iterations,

$$x_t = x_{t-1} - \eta\left(f_{i_t}'(x_{t-1}) - f_{i_t}'(x^s) + \mu^s\right).$$

- SVRG 'outer' update sets μ^s:
  - Set $x^s = x_t$ or random x_t since last update.
  - Set $\mu^s = f'(x^s) = \frac{1}{n}\sum_{i=1}^n f_i'(x^s)$ (full gradient).
- Convergence rate depends on $\rho(a,b) \triangleq \frac{1}{1-2\eta a}\left(\frac{1}{m\mu\eta} + 2b\eta\right)$
- SVRG achieves linear convergence rate (faster than sublinear rate of SG),

$$\mathbb{E}[f(x^{s+1}) - f(x^*)] \le \rho(L,L)\mathbb{E}[f(x^s) - f(x^*)],$$

Convergence Rate with Noise

- Consider using $\mu^s = f'(x^s) + e^s$, where e^s is an error term.
- We show that (assuming $\|x_t - x^*\| \le Z$)

$$\mathbb{E}[f(x^{s+1}) - f(x^*)] \le \rho(L,L)\mathbb{E}[f(x^s) - f(x^*)] + \frac{Z\mathbb{E}\|e^s\| + \eta\mathbb{E}\|e^s\|^2}{1-2\eta L}.$$

  - Error does not slow down SVRG when far from the solution.
  - If error converges to zero linearly, we maintain linear convergence rate.
- Sampling proportional to Lipschitz constant L_i of each f_i' gives faster rate.

Batching SVRG

- We can approximate μ^s with fewer gradients by using a subset $\mathcal{B}^s$ of the f_i'.
- If variance of gradient norms is bounded

$$\frac{1}{n-1}\sum_{i=1}^{n}\left[\|f_i'(x^s)\|^2 - \|f'(x^s)\|^2\right] \le S^2,$$

then we can bound expected size of error e^s,

$$\mathbb{E}\|e^s\|^2 \le \frac{n-|\mathcal{B}^s|}{n|\mathcal{B}^s|}S^2.$$

- To achieve a linear rate it is sufficient to have

$$|\mathcal{B}^s| \ge \frac{nS^2}{S^2 + n\gamma\bar{\rho}^{2s}},$$

for some γ and ρ (increase batch size exponentially until $n/2$, then more slowly).

Algorithm 1 Batching SVRG

Input: initial vector x^0, update frequency m, learning rate η.
for $s = 0,1,2,\ldots$ **do**
  $\mathcal{B}^s = |\mathcal{B}^s|$ elements sampled without replacement from $\{1,2,\ldots,n\}$.
  $\mu^s = \frac{1}{|\mathcal{B}^s|}\sum_{i\in\mathcal{B}^s} f_i'(x^s)$
  $x_0 = x^s$
  for $t = 1,2,\ldots,m$ **do**
    Randomly pick $i_t \in 1,\ldots,n$
    $x_t = x_{t-1} - \eta(f_{i_t}'(x_{t-1}) - f_{i_t}'(x^s) + \mu^s)$ $(*)$
  end for
  set $x^{s+1} = x_m$
end for

- Mixed SG and SVRG method: use regular SG update in $(*)$ if i_t is not in $\mathcal{B}^s$.
  - Starts out doing regular SG and slowly adds variance reduction.
  - Early iterations only require 1 gradient evaluation.
- Using $\alpha = |\mathcal{B}^s|/n$ and assuming $\mathbb{E}\|f_i'(x)\|^2 \le \sigma^2$, achieves faster rate if σ^2 small,

$$\mathbb{E}[f(x^{s+1}) - f(x^*)] \le \rho(L,\alpha L)\mathbb{E}[f(x^s) - f(x^*)] + \frac{Z\mathbb{E}\|e^s\| + \eta\mathbb{E}\|e^s\|^2 + \frac{\eta\sigma^2}{2}(1-\alpha)}{1-2\eta L},$$

Using Support Vectors

- Consider objectives like the Huberized hinge loss,

$$\min_{x\in\mathbb{R}^d}\frac{1}{n}\sum_{i=1}^{n} f(b_i a_i^T x), \quad f(\tau) = \begin{cases} 0 & \text{if } \tau > 1+\epsilon, \\ 1-\tau & \text{if } \tau < 1-\epsilon, \\ \frac{(1+\epsilon-\tau)^2}{4\epsilon} & \text{if } |1-\tau| \le \epsilon, \end{cases}$$

which is differentiable but many gradients are zero at solution.
- Skip evaluating gradient at exponentially-increasing interval if it remains at zero.

Algorithm 2 Heuristic for skipping evaluations of f_i at x

if $sk_i = 0$ **then**
  compute $f_i'(x)$.
  if $f_i'(x) = 0$ **then**
    $ps_i = ps_i + 1$.   {Update the number of consecutive times $f_i'(x)$ was zero.}
    $sk_i = 2^{\max\{0, ps_i - 2\}}$.   {Skip exponential number of future evaluations if it remains zero.}
  else
    $ps_i = 0$.   {This could be a support vector, do not skip it next time.}
  end if
  return $f_i'(x)$.
else
  $sk_i = sk_i - 1$.   {In this case, we skip the evaluation.}
  return 0.
end if

Regularized and Mini-batched SVRG

- Consider optimizing simple plus sum of smooth functions,

$$\min_{x\in\mathbb{R}^d} f(x) \equiv h(x) + \frac{1}{n}\sum_{i=1}^{n} g_i(x).$$

- Non-smooth h: we analyze proximal-SVRG method with error.
- Smooth h: we consider SVRG-like iteration that uses exact gradient of h,

$$x_{t+1} = x_t - \eta\left(h'(x_t) + g_{i_t}'(x_t) - g_{i_t}'(x^s) + \mu^s\right),$$

where $\mu^s = \frac{1}{n}\sum_{i=1}^n g_i(x^s)$.
- Common example is $h(x) = \frac{\lambda}{2}\|x\|^2$ and using the update

$$x_{t+1} = (1-\eta\lambda)x_t - \eta\left(g_{i_t}'(x_t) - g_{i_t}'(x^s) + \mu^s\right),$$

which is appealing if gradients g_i are sparse.
- Using $L_m = \max\{L_g, L_h\}$, this achieves a faster rate of

$$\mathbb{E}[f(x^{s+1}) - f(x^*)] \le \rho(L_m, L_m)\mathbb{E}[f(x^s) - f(x^*)],$$

- We consider mini-batch where h is f_i whose gradients have largest Lipschitz constants.
- We also explored mini-batches based on function value and gradient norms.

Learning Efficiency

- Bottou & Bousquet show that we can write generalization error $\mathcal{E}$ using three terms

$$\mathcal{E} = \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}}.$$

- We analyze algorithms like SAG and SVRG under their assumptions.
- Methods like SAG and SVRG can obtain better bounds in certain settings.

Algorithm	Time to reach $\mathcal{E}_{\text{opt}} \le \epsilon$	Time to reach $\mathcal{E} = O(\mathcal{E}_{app} + \epsilon)$	Previous with $\kappa \sim n$
FG	$\mathcal{O}\left(n\kappa d\log\left(\frac{1}{\epsilon}\right)\right)$	$\mathcal{O}\left(\frac{d^2\kappa}{\epsilon^{1/\alpha}}\log^2\left(\frac{1}{\epsilon}\right)\right)$	$\mathcal{O}\left(\frac{d^3}{\epsilon^{2/\alpha}}\log^3\left(\frac{1}{\epsilon}\right)\right)$
SG	$\mathcal{O}\left(\frac{d\nu\kappa^2}{\epsilon}\right)$	$\mathcal{O}\left(\frac{d\nu\kappa^2}{\epsilon}\right)$	$\mathcal{O}\left(\frac{d^3\nu}{\epsilon}\log^2\left(\frac{1}{\epsilon}\right)\right)$
SVRG	$\mathcal{O}\left((n+\kappa)d\log\left(\frac{1}{\epsilon}\right)\right)$	$\mathcal{O}\left(\frac{d^2}{\epsilon^{1/\alpha}}\log^2\left(\frac{1}{\epsilon}\right) + \kappa d\log\left(\frac{1}{\epsilon}\right)\right)$	$\mathcal{O}\left(\frac{d^2}{\epsilon^{1/\alpha}}\log^2\left(\frac{1}{\epsilon}\right)\right)$

Experiment Results (Logistic Regression and Huberized SVM)