Stop Wasting My Gradient: A Practical SVRG

Reza Babanezhad Harikandeh
Joint work with: Mark Schmidt (UBC), Mohamed Ahmed (UBC), Jakub Konecny (University of Edinburgh)

University of British Columbia
rezababa@cs.ubc.ca

WCOM
Oct, 2016
We want to minimize the sum of a finite set of smooth functions

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

We are interested in cases where \( n \) is very large.

We will focus on strongly-convex functions

Simplest example is \( l_2 \)-regularized least-squares

\[
f_i(x) = (a_i^T x - b_i)^2 + \frac{\lambda}{2} \| x \|^2
\]

Common framework in data fitting problem
- logistic regression, Huber regression, smooth SVMs, CRFs, etc.
Deterministic gradient method [Cauchy, 1847]:

\[ X_{t+1} = X_t - \alpha_t f'(X_t) = X_t - \frac{\alpha_t}{n} \sum_{i=1}^{n} f'_i(X_t) \]

- Linear convergence rate
- Iteration cost is linear in n
Stochastic vs. Deterministic Gradient Methods

- Deterministic gradient method [Cauchy, 1847]:
  \[ X_{t+1} = X_t - \alpha_t f'(X_t) = X_t - \frac{\alpha_t}{n} \sum_{i=1}^{n} f'_i(X_t) \]
  - Linear convergence rate
  - Iteration cost is linear in \( n \)
- Stochastic gradient method [Robins and Monro, 1951]:
  Randomly pick \( i_t \) in iteration \( t \) from \( \{1, \ldots, n\} \)
  \[ X_{t+1} = X_t - \alpha_t f'_{i_t}(X_t) \]
  - Iteration cost is independent of \( n \)
  - Sub-linear convergence rate
Motivations for New Methods

- **Stochastic Variance Reduced Methods**: Linear convergence rate + $O(1)$ iteration cost

![Graph showing log(excess cost) vs time with stochastic and deterministic lines.](image)
Motivations for New Methods

- SAG [Le Roux et al. 2012]
- SDCA [Shalev-Shwartz and Zhang, 2013]
- MISO [Marial, 2013]
- SAGA [Defazio, et al., 2014]
- These methods all need memory to store gradient of $f_i$’s or dual variable
  - $O(nd)$ space for general objective function.
Recent methods with similar rates that avoid memory:
- Mixed Gradient [Mahdavi & Jin, 2013, Zhang et al., 2013]
- Stochastic variance-reduced gradient (SVRG) [Johnson & Zhang, 2013]
- Semi-stochastic gradient [Konecny & Richtarik, 2013]

Memory is only $O(d)$, but they require extra gradient calculations:
- Two gradients on each iteration.
- Occasional calculation of all $n$ gradients.

Extra calculations make them slower than SAG and friends.
1. Deterministic, stochastic, and finite-sum methods
2. Wasting fewer gradients in SVRG
3. Some Heuristic For SVM
4. Conclusion
SVRG Algorithm\( (m, \alpha, x_0) \)

- start with \( x_0 \)
  - for \( t = 0, 1, \ldots, m \)
    - randomly pick \( i_t \)
    \[
x^{t+1} = x^t - \alpha(f'_t(x^t))
    \]
SVRG Algorithm( $m, \alpha, x_0$)

- start with $x_0$
  - for $t = 0, 1, \ldots, m$
    - randomly pick $i_t$
      \[ x^{t+1} = x^t - \alpha(f_{i_t}'(x^t) - f_{i_t}'(x_s) + d_s) \] (two gradients per iteration)
SVRG Algorithm ($m, \alpha, x_0$)

- start with $x_0$
- for $s = 0, 1, 2, \ldots$ (outer loop)
  
  
  $d_s = \frac{1}{n} \sum_{i=1}^{n} f'_i(x_s)$ (full gradient evaluation)

- $x^0 = x_s$
- for $t = 0, 1, \ldots, m$ (inner loop)
  
  randomly pick $i_t$
  
  $x^{t+1} = x^t - \alpha(f'_i(x^t) - f'_i(x_s) + d_s)$ (two gradients per iteration)

- $x_{s+1} = x^t$ for a random $t \in \{1, \ldots, m\}$
Convergence Analysis of SVRG

Assumptions:
- Each $f_i$ is convex.
- Each $\nabla f_i$ is $L$-Lipschitz continuous.
- $f$ is $\mu$-strongly convex.

Johnson & Zhang [2013] show that outer loop satisfies
\[
\mathbb{E}[f(x_{s+1}) - f(x^*)] \leq \rho[f(x_s) - f(x^*)], \quad \rho = \frac{1}{1 - 2\alpha L} \left(2L\alpha + \frac{1}{m\mu\alpha}\right)
\]
SVRG rate is very fast for appropriate step size $\alpha$ and inner-loop size $m$.

In practice: $m = n$, $\alpha = 1/L$, $x_{s+1} = x^m$
Convergence Analysis of SVRG with Error

Assume:
1. We approximate full gradient by \( d^s = f'(x_s) + e^s \)
2. \( \|x^t - x^*\| \leq Z \) for some \( Z \)

Then SVRG with error satisfies

\[
E[f(x_{s+1}) - f(x^*)] \leq \rho [f(x_s) - f(x^*)] + \frac{\alpha E[\|e^s\|^2] + Z E[\|e^s\|]}{1 - 2\alpha L}
\]

Implications
1. Faster rate when far from solution.
2. Same convergence rate if \( \max\{E[\|e^s\|], E[\|e^s\|^2]\} = O(\tilde{\rho}^s) \) for \( \tilde{\rho}^s \leq \rho \)
Reducing Gradient Evaluations with Batching

- SVRG requires $2m + n$ gradients for each $m$ iterations.
- We can reduce the $n$ by using a mini-batch $\mathcal{B}^s$ of training examples

$$d^s = \frac{1}{|\mathcal{B}^s|} \sum_{i \in \mathcal{B}^s} f_i'(x_s)$$

- Special case of SVRG with error, batch size controls error.

$$|\mathcal{B}^s| \geq \frac{nS^2}{S^2 + n\gamma \tilde{\rho}^2s}$$

[Aravkin et al, 2012]
Algorithm 1 Batching SVRG

**Input:** initial vector $x^0$, update frequency $m$, learning rate $\alpha$.

**for** $s = 0, 1, 2, \ldots$ **do**

- $B^s = |B^s|$ elements sampled without replacement from $\{1, 2, \ldots, n\}$.
- $d^s = \frac{1}{|B^s|} \sum_{i \in B^s} f'_i(x^s)$
- $x^0 = x^s$

**for** $t = 1, 2, \ldots, m$ **do**

- Randomly pick $i_t \in 1, \ldots, n$
- $x^{t+1} = x^t - \alpha(f'_{i_t}(x^t) - f'_{i_t}(x^s) + d^s)$

**end for**

**option I:** set $x_{s+1} = x^m$

**option II:** set $x_{s+1} = x^t$ for a random $t \in \{1, \ldots, m\}$

**end for**

- Growing-batch reduces $n$ in the $2m + n$ cost of SVRG.
- But does not improve the 2
- Mixing SGD with SVRG
Numerical Experiments with Batching

Training/testing loss for $\ell_2$-regularized logistic on spam filtering data.

$$\arg \min_{x \in \mathbb{R}^d} \frac{\lambda}{2} \|x\|^2 + \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-b_i a'_i x))$$
Outline

1. Deterministic, stochastic, and finite-sum methods
2. Wasting fewer gradients in SVRG
3. Some Heuristic For SVM
4. Conclusion
Identifying Support Vectors

- Mixed strategy improves error when far from solution.
- For certain objectives, can improve close to solution.
- Consider Huberized hinge loss problem [Rosset & Zhu, 2006]

\[
\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f(b_i a'_i x),
\]

\[
f(\tau) = \begin{cases} 
0 & \text{if } \tau > 1 + \epsilon \\
1 - \tau & \text{if } \tau < 1 - \epsilon \\
\frac{(1+\epsilon-\tau)^2}{4\epsilon} & \text{if } |1 - \tau| \leq \epsilon 
\end{cases}
\]

- The solution is sparse in the \( f'_i \) (has support vectors).
Non-support examples do not contribute to solution

We can skip gradient evaluations where we expected/know that $f'_i = 0$

Approach 1: sound pruning
- Maintain list of support vectors at $x_s$.
- Do not evaluate $f_i(x_s)$ if it is not a support vector.
- Can reduce number of gradients per iteration to 1.
Using Support Vectors

- Non-support examples do not contribute to solution
- We can skip gradient evaluations where we expected/know that $f_i' = 0$
- **Approach 2: heuristic pruning**
  - Keep track of the number of times we $f'_i(x_s) = 0$ or $f'_i(x^t) = 0$.
  - If it continues to be zero, skip its next 2 evaluations.
  - If it continues to be zero, skip its next 4 evaluations.
  - If it continues to be zero, skip its next 8 evaluations.
  - Can reduce number of gradients per iteration to 1 exponentially.
\( \mathcal{L}_2 \)-regularized Huberized hinge on spam filtering data.
1. Deterministic, stochastic, and finite-sum methods
2. Wasting fewer gradients in SVRG
3. Some Heuristic For SVM
4. Conclusion
Conclusion

- Stochastic methods for minimizing finite sum with linear convergence
- SVRG is the only method without a memory requirement
- Reducing gradient evaluation by inexact full gradient
- A heuristic SVM algorithm
- Other variants and analysis
  - Mixed Strategy
  - Proximal SVRG
  - SVRG with non-uniform sampling
  - Fixed-Random Mini-Batching Strategy
  - Generalization error
- Thank you!