One Method to Rule Them All: Variance Reduction for Data, Parameters and Many New Methods

Filip Hanzely, Peter Richtarik
Problem and Introduction
Problem

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

Matrix smoothness \((\text{specified later})\)

\(M_i\) smooth and convex

Strongly convex

Convex, proximable \((\text{specified later})\)

Oracle: random linear measurements of (stochastic) gradient

Main result: Very general algorithm
Problem

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

\(M_i\) smooth and convex

\[f_i(x) \leq f_i(y) + \langle \nabla f_i(y), x - y \rangle + \frac{1}{2} (x - y)^\top M_i (x - y)\]

\(\phi_i\) is \(L_i\) smooth

\[f_i(x) = \phi_i(A_i x)\]

\(M_i = L_i A_i^\top A_i\)
Gradient Descent, SGD and Variance reduction

\[ GD: \quad x^+ = x - \alpha \nabla f(x) \]
\[ SGD: \quad x^+ = x - \alpha \nabla f_i(x) \]
\[ \text{SAGA: } x^+ = x - \alpha g \]

Variance reduced stochastic gradient
\[ g \to 0 \text{ as } x \to x^*, \quad E[g] = \nabla f(x) \]
Variance Reduction in Optimization: Classical Approach
Control Variates in optimization [Johnson and Zhang 2013, Defazio et al. 2014]

Trick to reduce variance of stochastic gradient

Standard in statistics, relatively new in optimization

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

(no \(\psi\) for simplicity)

\[
x^{k+1} = x^{k} - \alpha g^k
\]

\[
g^k = \nabla f_i (x^k) - Y^k + E[Y^k]
\]

Goal: High Cov\((Y^k, \nabla f_i(x^k))\) (to decrease variance of \(g^k\))

Solution: Set \(Y^k = \nabla f_i (\phi_i^k)\) for \(\phi_i^k = x^l\), where \(l\) is index corresponding to the last evaluation of \(\nabla f_i(\cdot)\)

As \(x^k \to x^*\), we will have both \(\nabla f_i (x^k) \to \nabla f_i (x^*)\) and \(Y^k \to \nabla f_i (x^*)\) and thus \(Y^k \to \nabla f_i (x^k)\), which means \(\text{Var}(g^k) \to 0\)
Algorithm 2 SAGA [3]

Require: learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$
Set $\psi_j^0 = x^0$ for each $j \in \{1, 2, \ldots, n\}$

for $k = 0, 1, 2, \ldots$ do
    Sample $j \in [n]$ uniformly at random
    Set $\phi_j^{k+1} = x^k$ and $\phi_i^{k+1} = \phi_i^k$ for $i \neq j$
    $g^k = \nabla f_j(\phi_j^{k+1}) - \nabla f_j(\phi_j^k) + \frac{1}{n} \sum_{i=1}^n \nabla f_i(\phi_i^k)$
    $x^{k+1} = \text{prox}_{\alpha \psi}(x^k - \alpha g^k)$

end for

Construct control variates from freshest information

stochastic gradient via control variates
Variance Reduction in Optimization: SEGA
Regularization and Proximal operator

\[
\text{minimize } f(x) + \psi(x) \quad \text{(no finite sum)}
\]

convex, non-smooth, simple

Proximal operator of \( \psi \) is computable

\[
\text{prox}_{\alpha \psi}(x) \overset{\text{def}}{=} \arg\min_{y \in \mathbb{R}^n} \left\{ \psi(y) + \frac{1}{2\alpha} \|y - x\|^2 \right\}
\]

If \( \psi \) is indicator function, then prox is projection
If \( \psi \) is L1 loss, then prox is soft thresholding

Proximal gradient: \( x^+ = \text{prox}_{\alpha \psi}(x - \alpha \nabla f(x)) \)

\[
= \arg\min_{y} \left\{ \psi(y) + \frac{1}{2\alpha} \|y - (x - \alpha \nabla f(x))\|^2 \right\}
\]

Nearly gradient step, but considers \( \psi \) as well
Coordinate descent and proximal regularizer

CD: Compute partial derivative, take step

Example
minimize $f(x) + \psi(x)$

$f(x) = \|x - z\|^2$

$\|z\|$ is big enough

$$\psi(x) = \begin{cases} 
0 & \|x\| \leq 1 \\
\infty & \|x\| > 1 
\end{cases}$$

Solution: CD with variance reduction

= SEGA

Proximal CD can jump away from optimum

CD oscillates!
SEGA [Hanzely et al. 2018]

Coordinate descent with variance reduction

\[
\min_{x \in \mathbb{R}^d} f(x) + \psi(x)
\]

\[
E[g^k] = \nabla f(x^k)
\]

\[
x^{k+1} = \prox_{\alpha \psi} (x^k - \alpha g^k) \quad g^k = d \nabla_j f(x^k) - Y^k + E[Y^k]
\]

Goal: High Cov\((Y^k, d \nabla_j f(x^k))\) (to decrease variance of \(g^k\))

Solution: Set \(Y^k = d \nabla_j f(\phi^k_j)\) for \(\phi^k_j = x^l\), where \(l\) is index corresponding to the last evaluation of \(\nabla_j f(\cdot)\).

As \(x^k \to x^*\), we will have both \(\nabla_j f(x^k) \to \nabla_j f(x^*)\) and \(Y^k \to d \nabla_j f(x^*)\) and thus \(\text{Var}(g^k) \to 0\)
SEGA [Hanzely et. al. 2018]

Coordinate descent with variance reduction

Construct control variates from freshest information

**Algorithm 5 SEGA**

**Require:** Stepsize $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$

Set $h^0 = 0$

for $k = 0, 1, 2, \ldots$ do

Sample $i \in \{1, \ldots, d\}$ uniformly at random

Set $h^{k+1} = h^k + e_i(\nabla_i f(x^k) - h_i^k)$

$g^k = h^k + d e_i(\nabla_i f(x^k) - h_i^k)$

$x^{k+1} = \text{prox}_{\alpha \psi}(x^k - \alpha g^k)$

end for

stochastic gradient via control variates
Algorithm 5 SEGA

Require: Stepsize $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$

Set $h^0 = 0$

for $k = 0, 1, 2, \ldots$ do

Sample $i \in \{1, \ldots, d\}$ uniformly at random

Set $h^{k+1} = h^k + e_i(\nabla_i f(x^k) - h_i^k)$

$g^k = h^k + d e_i(\nabla_i f(x^k) - h_i^k)$

$x^{k+1} = \text{prox}_{\alpha \psi}(x^k - \alpha g^k)$

end for

$\psi = 0$: Up to constant factor rate of coordinate descent

$\psi \neq 0$: No importance sampling

$\psi = 0$: Up to constant factor rate of accelerated coordinate descent
Variance Reduction in Optimization: Parallel Coordinate Descent
Parallel Coordinate Descent

Workers: Compute the local gradient update and send it to PS

Server: Average updates and send them back to workers

Bottleneck: Communication from workers to server
Parallel Coordinate Descent

Holds for GD, SGD, SAGA, ASGD
if $\forall i : \nabla f_i(x^*) = 0$

Holds for GD with SEGA trick (no extra assumptions)

[Mishchenko, Hanzely and Richtarik, 2018]
All workers might throw away as much as $\frac{n-1}{n}$ of total number of coordinates independently at random without hurting the convergence

Constant factor at most

Reduced communication to the server for no cost!

Natural Extension: Can we do SAGA and SEGA at the same time?

Holds for GD with SEGA trick (no extra assumptions)
Unification of the Methods
Goals

Unification of broad range of variance reduced algorithms in single method and single analysis

Analysis that include best bounds for recovered algorithms

Understanding relations between algorithms

New algorithms

Arbitrary sampling results
(tight rate for given smoothness structure of objective and algorithm)

Proximal methods
High level idea

Jacobian
\[ \mathbf{G}(x) = [\nabla f_1(x), \ldots, \nabla f_n(x)] \in \mathbb{R}^{d \times n} \]

Oracle: random linear measurement of Jacobian \( \mathcal{S}\mathbf{G}(x) \)

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

random linear operator (fixed distribution)

Goal: Develop efficient variance reduction
Provide as fast rate as possible
3 key ingredients

1) Construct Jacobian estimator for control variates (sketch and project)

\[ J^{k+1} = \arg\min_J \| J - J^k \|^2 \quad \text{s. t. } S J = S G (x^k) \]

random linear operator

\[ g^k = \frac{1}{n} J^k e + \frac{1}{n} U (G (x^k) - J^k) e \]

random linear operator; \( E[U(X)] = X \)

S, U might be correlated

2) Construct stochastic gradient using control variates

3) Take proximal SGD step
Algorithm

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

**Algorithm 1** Generalized JacSketch (GJS)

1. **Parameters:** Step size \( \alpha > 0 \), random projector \( \mathcal{S} \) and unbiased sketch \( \mathcal{U} \)
2. **Initialization:** Choose solution estimate \( x^0 \in \mathbb{R}^d \) and Jacobian estimate \( \mathbf{J}^0 \in \mathbb{R}^{d \times n} \)
3. for \( k = 0, 1, \ldots \) do
4. Sample realizations of \( \mathcal{S} \) and \( \mathcal{U} \), and perform sketches \( SG(x^k) \) and \( UG(x^k) \)
5. \( \mathbf{J}^{k+1} = \mathbf{J}^k - \mathcal{S} (\mathbf{J}^k - G(x^k)) \)
6. update the Jacobian estimate
   \( g^k = \frac{1}{n} \mathbf{J}^k e + \frac{1}{n} \mathcal{U} (G(x^k) - \mathbf{J}^k) e \)
7. construct the gradient estimator
   \( x^{k+1} = \text{prox}_{\alpha \psi} (x^k - \alpha g^k) \)
8. perform the proximal SGD step
9. end for
Convergence (main theorem)

Property of randomness and smoothness

\[
\mathcal{M}^{\frac{1}{2}} : \mathbb{R}^{d \times n} \to \mathbb{R}^{d \times n}; \mathcal{M}^{\frac{1}{2}}(X) = [M^{\frac{1}{2}}_{1}(X_{1}), M^{\frac{1}{2}}_{2}(X_{2}), \cdots, M^{\frac{1}{2}}_{n}(X_{n})]
\]

Smoothness operator (notation)

\[
\text{Condition on the stepsize } \alpha
\]

\[
\frac{2\alpha}{n^2} \mathbb{E}\left[\|UXe\|^2\right] + \left\|(\mathbb{E}[S])^{\frac{1}{2}}BM^{\frac{1}{2}}X\right\|^2 \leq \frac{1}{n} \left\|\mathcal{M}^{\frac{1}{2}}X\right\|^2
\]

\[
\frac{2\alpha}{n^2} \mathbb{E}\left[\|UXe\|^2\right] + \left\|\left(I - \mathbb{E}[S]\right)^{\frac{1}{2}}BM^{\frac{1}{2}}X\right\|^2 \leq (1 - \alpha \sigma) \left\|BM^{\frac{1}{2}}X\right\|^2
\]

Satisfied for small \( \alpha \)

\[\mathbb{E}\left[\Psi^k\right] \leq (1 - \alpha \sigma)^k \Psi^0\]

For all \( X \)

Strong convexity

\[
\Psi^k := \|x^k - x^*\|^2 + \alpha \left\|BM^{\frac{1}{2}}\left(J^k - G(x^*)\right)\right\|^2
\]

Maximize \( \alpha \) s. t. the bounds hold
Special cases
 Saga with arbitrary sampling

Algorithm 3 SAGA with arbitrary sampling (a variant of (Qian et al., 2019))

Require: learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, random sampling $R \subseteq \{1, 2, \ldots, n\}$

Set $\phi_j^0 = x^0$ for each $j \in [n]$

for $k = 0, 1, 2, \ldots$ do

Sample random $R^k \subseteq \{1, 2, \ldots, n\}$

Set $\phi_j^{k+1} = \begin{cases} x^k & j \in R^k \\ \phi_j^k & j \not\in R^k \end{cases}$

$g^k = \frac{1}{n} \sum_{j=1}^{n} \nabla f_j(x^k_j) + \sum_{j \in R^k} \frac{1}{np_j} \left( \nabla f_j(x_j^{k+1}) - \nabla f_j(x_j^k) \right)$

$x^{k+1} = \text{prox}_{\alpha \psi}(x^k - \alpha g^k)$

end for

$p_j = \text{Prob}(j \in S')$

Corollary 13 (Convergence rate of SAGA) Let $\alpha = \min_{j} \frac{np_j}{4v_j + n\sigma}$. Then the iteration complexity of Algorithm 3 is $\max_j \left( \frac{4v_j + n\sigma}{n\sigma p_j} \right) \log \frac{1}{\epsilon}$.

$\forall h_i \in \mathbb{R}^d : \mathbb{E} \left[ \left\| \sum_{i \in S} M_{i}^{\frac{1}{2}} h_i \right\|^2 \right] \leq \sum_{i=1}^{n} p_i v_i \| h_i \|^2$

Recovered rate from Qian et al. (2019) and Defazio et al. (2014)
Saga with arbitrary sampling

\[
\mathbb{E} \left[ \left\| \sum_{i \in S} \frac{1}{2} M_i h_i \right\|^2 \right] \leq \sum_{i=1}^{n} p_i v_i \| h_i \|^2
\]

What is this???

\[
|S| = 1 \text{ w. p. 1} \quad \rightarrow \quad v_i = \lambda_{\max} M_i
\]

\[
|S| = n \text{ w. p. 1} \quad \rightarrow \quad v_i = \lambda_{\max} \left( \sum_{i=1}^{n} M_i \right)
\]

\[
P_{i,j} = \text{Prob}(i \in S, j \in S) ee^\top; P \in \mathbb{R}^{nd \times nd}
\]

\[
P \circ \left( \left( \left( M_1^{\frac{1}{2}}, \ldots, M_n^{\frac{1}{2}} \right) \left( M_1^{\frac{1}{2}}, \ldots, M_n^{\frac{1}{2}} \right)^\top \right) \right) \preceq \text{Diag}(p \circ v) \otimes I
\]

Same story as for coordinate descent

\[
P \circ M \preceq \text{Diag}(p \circ v)
\]

Use minibatch samplings for CD
Saga with arbitrary sampling

**Corollary 13 (Convergence rate of SAGA)** Let $\alpha = \min_j \frac{np_j}{4v_j n\sigma}$. Then the iteration complexity of Algorithm 3 is $\max_j \left( \frac{4v_j + n\sigma}{n\sigma p_j} \right) \log \frac{1}{\epsilon}$.

$$L_j = \lambda_{\max} M_j$$
$$\bar{L} = \frac{1}{n} \sum_{i=1}^{n} L_i$$

$f_j$ is $L_j$ smooth; single element with $p \propto L \Rightarrow \frac{4\bar{L}}{\mu} + \max_j p_j^{-1}$

$f$ is $L$ smooth; sample all w.p. 1 $\Rightarrow \frac{4L}{\mu}$

$L = \lambda_{\max} \left( \frac{1}{n} \sum_{i=1}^{n} M_i \right)$

$L \leq \bar{L} \leq nL$

+ everything in between
Loopless SVRG with arbitrary sampling

Algorithm 8 LSVRG (LSVRG (Kovalev et al., 2019) with arbitrary sampling) [NEW METHOD]

Require: learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, random sampling $R \subseteq \{1, 2, \ldots, n\}$

Set $\phi = x^0$

for $k = 0, 1, 2, \ldots$ do

Sample a random subset $R^k \subseteq \{1, 2, \ldots, n\}$

$g^k = \frac{1}{n} \sum_{j=1}^{n} \nabla f_j(\phi^k) + \sum_{j \in R^k} \frac{1}{np_j} (\nabla f_j(x^k) - \nabla f_j(\phi^k))$

$x^{k+1} = \text{prox}_{\alpha \psi}(x^k - \alpha g^k)$

Set $\phi^{k+1} = \begin{cases} 
    x^k & \text{with probability } \rho \\
    \phi^k & \text{with probability } 1 - \rho 
\end{cases}$

end for

Corollary 22 (Convergence rate of LSVRG) Let $\alpha = \min_j \frac{v_i}{4p_j} \frac{n}{\sigma}$. Then, iteration complexity of Algorithm 8 is $\max_j \left(4 \frac{v_i}{n\sigma p_j} + \frac{1}{\rho} \right) \log \frac{1}{\epsilon}$.

$f_j$ is $L_j$ smooth; single element with $p \propto L \Rightarrow \frac{4\bar{L}}{\mu} + \rho^{-1}$

New results!
Shifted SGD with arbitrary sampling

If $G(x^*)$ is known – no need for learning Jacobian

**Algorithm 7 SGD-shift [NEW METHOD]**

**Require:** learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, random sampling $R \subseteq \{1, 2, \ldots, n\}$

for $k = 0, 1, 2, \ldots$ do

Sample random $R^k \subseteq \{1, 2, \ldots, n\}$

$g^k = \frac{1}{n} G(x^*) e + \sum_{j \in R^k} \frac{1}{np_j} (\nabla f_j(x^k) - \nabla f_j(x^*))$

$x^{k+1} = \text{prox}_{\alpha \psi}(x^k - \alpha g^k)$

end for

**Corollary I.1** (Convergence rate of SGD-AS-shift). Suppose that $f_j$ is $M_j$-smooth for all $j$ and suppose that $\nu$ satisfies (32). Let $\alpha = n \min_j \frac{\nu_j}{v_j}$. Then, iteration complexity of Algorithm 7 is

$$\max_j \left( \frac{v_j}{np_j \sigma} \right) \log \frac{1}{\epsilon}.$$
SEGA with arbitrary sampling

Algorithm 5 SEGA with arbitrary sampling

Require: Stepsize $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, random sampling $L \subseteq \{1, 2, \ldots, d\}$

Set $h^0 = 0$

for $k = 0, 1, 2, \ldots$ do

Sample random $L^k \subseteq \{1, 2, \ldots, d\}$

Set $h^{k+1} = h^k + \sum_{i \in L^k} (\nabla_i f(x^k) - h^k_i) e_i$

$g^k = h^k + \sum_{i \in L^k} \frac{1}{p_i} (\nabla_i f(x^k) - h^k_i) e_i$

$x^{k+1} = \text{prox}_{\alpha \psi}(x^k - \alpha g^k)$

end for

Corollary 16 (Convergence rate of SEGA) Iteration complexity of Algorithm 5 with $\alpha = \min_i \frac{p_i}{4m_i + \sigma}$ is $\max_i \left( \frac{4m_i + \sigma}{p_i \sigma} \right) \log \frac{1}{\epsilon}$.

$M = \text{Diag}(m)$. Tighter rate than from original SEGA paper
SVRCD with arbitrary sampling

**Algorithm 6 SVRCD [NEW METHOD]**

**Require**: starting point $x^0 \in \mathbb{R}^d$, random sampling $L \subseteq \{1, 2, \ldots, d\}$, probability $\rho$, stepsize $\alpha > 0$

Set $h^0 = 0$

for $k = 0, 1, 2, \ldots$ do

Sample random $L^k \subseteq \{1, 2, \ldots, d\}$

$g^k = h^k + \sum_{i \in L^k} \frac{1}{p_i}(\nabla_i f(x^k) - h_i^k)e_i$

$x^{k+1} = \text{prox}_{\alpha \psi}(x^k - \alpha g^k)$

Set $h^{k+1} = \begin{cases} 
  h^k & \text{with probability } 1 - \rho \\
  \nabla f(x^k) & \text{with probability } \rho 
\end{cases}$

end for

**Corollary 18** Iteration complexity of Algorithm 6 with $\alpha = \min_i \frac{1}{4m_i/p_i + \sigma/\rho}$ is $\left(\frac{1}{\rho} + \max_i \frac{4m_i}{p_i \sigma}\right) \log \frac{1}{\epsilon}$

New Algorithm!

$M = \text{Diag}(m)$. 

31
[Mishchenko, Hanzely and Richtarik, 2018]

All workers might throw away as much as \( \frac{n-1}{n} \) of total number of coordinates independently at random without hurting the convergence

Reduced communication to the server for no cost! Constant factor at most

Natural Extension: Can we do SAGA and SEGA at the same time?

YES! With arbitrary sampling
Algorithm 16 ISAEGA [NEW METHOD]

**Input:** $x^0 \in \mathbb{R}^d$, # parallel units $T$, each owning set of indices $N_t$ (for $1 \leq t \leq T$), distributions $\mathcal{D}_t$ over subsets of $N_t$, distributions $\mathcal{D}_t$ over subsets coordinates $[d]$, stepsize $\alpha$

$J^0 = 0$

for $k = 0, 1, \ldots$ do

    for $t = 1, \ldots, T$ in parallel do

        Sample $R_t \sim \mathcal{D}_t; R_t \subseteq N_t$ (independently on each machine)
        Sample $L_t \sim \mathcal{D}_t; L_t \subseteq [d]$ (independently on each machine)
        Observe $\nabla_{L_t} f_j(x^k)$ for $j \in R_t$

        For $i \in [d], j \in N_t$ set $J^{k+1}_{i,j} = \begin{cases} 
\nabla_i f_j(x^k) & \text{if } i \in [d], j \in R_t, i \in L_t \\
J^k_{i,j} & \text{otherwise}
\end{cases}$

        Send $J^{k+1}_{:N_t} - J^k_{:N_t}$ to master \quad \triangleright\text{Sparse; low communication}

    end for

    $g^k = \left( J^k + \sum_{t=1}^T \left( p^{t-1} p^t \right) \circ (\sum_{i \in L_t} e_i e_i^\top) (J^{k+1} - J^k)_{:N_t} \left( \sum_{j \in R_t} e_j e_j^\top \right) \right) e$

    $x^{k+1} = \text{prox}_{\alpha \psi}(x^k - \alpha g^k)$

end for
Other results
SAGA is a special case of SEGA

Equivalent problems

Apply SAGA

Equivalent methods

Apply SEGA

Theory of SEGA (with a twist) generalizes theory of SAGA
SVRG is a special case of SVRCD

Equivalent problems

Apply LSVRG

Equivalent methods

Apply SVRCD

Theory of SVRCD (+ small twist) generalizes theory of LSVRG

Same result for **Accelerated SVRCD** and Katyusha

new