CentralVR: Efficient Distributed SGD with Variance Reduction

Soham De
University of Maryland

Joint work with Tom Goldstein

Dec 15, 2016
massive model fitting

minimize \[ f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \]
MASSIVE MODEL FITTING

\[
\text{minimize } \quad f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)
\]

least squares

\[
\text{minimize } \quad \frac{1}{2} \| Ax - b \|^2 = \sum_{i} \frac{1}{2} (a_i x - b_i)^2
\]

Lots of other examples: SVM, Logistic Regression, Neural Networks
MASSIVE MODEL FITTING

minimize  \( f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \)

least squares

minimize  \( \frac{1}{2} \|Ax - b\|^2 = \sum_i \frac{1}{2} (a_i x - b_i)^2 \)

Lots of other examples: SVM, Logistic Regression, Neural Networks
SGD

approximate gradient

\[ \nabla f \approx \nabla f_k(x) \]
SGD

approximate gradient

\[ \nabla f \approx \nabla f_k(x) \]

stochastic gradient descent

\[ x^{k+1} = x^k - \tau_k \nabla f_k(x^k) \]
SGD

approximate gradient

\[ \nabla f \approx \nabla f_k(x) \]

stochastic gradient descent

\[ x^{k+1} = x^k - \tau_k \nabla f_k(x^k) \]

big error

solution improves
SGD

approximate gradient

$$\nabla f \approx \nabla f_k(x)$$

stochastic gradient descent

$$x^{k+1} = x^k - \tau_k \nabla f_k(x^k)$$
SGD

approximate gradient
\[ \nabla f \approx \nabla f_k(x) \]

stochastic gradient descent
\[ x^{k+1} = x^k - \tau_k \nabla f_k(x^k) \]

Error must decrease as we approach solution
\[ \lim_{k \to \infty} \tau_k = 0 \]
SGD

approximate gradient

$$\nabla f \approx \nabla f_k(x)$$

stochastic gradient descent

$$x^{k+1} = x^k - \tau_k \nabla f_k(x^k)$$

Error must decrease as we approach solution

$$\lim_{k \to \infty} \tau_k = 0$$

slow (sub-linear) convergence
SCALING SGD

Huge data distributed over clusters $\implies$ SGD *doesn't* scale
SCALING SGD

Huge data distributed over clusters → SGD doesn’t scale

Problems
SCALING SGD

Huge data distributed over clusters → SGD doesn’t scale

Problems

• Inherently sequential
SCALING SGD

Huge data distributed over clusters $\rightarrow$ SGD \textit{doesn’t} scale

Problems

• Inherently sequential
• Reduced benefits when \#nodes $> 100$
SCALING SGD

Huge data
distributed over clusters

⇒ SGD doesn’t scale

Problems

• Inherently sequential
• Reduced benefits when #nodes > 100
• Unstable with infrequent communication
IN THIS PAPER...

CentralVR

Leverages **Variance Reduction** techniques to speed up SGD
IN THIS PAPER...

CentralVR

Leverages **Variance Reduction** techniques to speed up SGD

- Can efficiently scale up to **hundreds** of distributed computing nodes
IN THIS PAPER...

CentralVR

Leverages Variance Reduction techniques to speed up SGD

• Can efficiently scale up to hundreds of distributed computing nodes

• Low communication costs suitable for large-scale heterogenous distributed environments
VARIANCE REDUCTION (VR)

\[ x^{k+1} = x^k - \tau \nabla f_k(x^k) \]
VARIANCE REDUCTION (VR)

\[ x^{k+1} = x^k - \tau \nabla f_k(x^k) \]

**What’s the error?**

\[ \text{error}_k = \nabla f_k(x_k) - \nabla f(x_k) \]
VARIANCE REDUCTION (VR)

\[ x^{k+1} = x^k - \tau \nabla f_k(x^k) \]

What’s the error?

\[ \text{error}_k = \nabla f_k(x_k) - \nabla f(x_k) \]

Corrected gradient descent

\[ x^{k+1} = x^k - \tau (\nabla f_k(x^k) - \text{error}_k) \]
VARIANCE REDUCTION (VR)

\[ x^{k+1} = x^k - \tau \nabla f_k(x^k) \]

**What’s the error?**

\[ \text{error}_k = \nabla f_k(x_k) - \nabla f(x_k) \]

Corrected gradient descent

\[ x^{k+1} = x^k - \tau(\nabla f_k(x^k) - \text{error}_k) \]

VR methods approximate this error term
Suppose sequential ordering over data indices

\[ x_m^k \]  :  \( k \)-th iterate of \( m \)-th epoch
Suppose sequential ordering over data indices

\[ x^k_m : k\text{-th iterate of } m\text{-th epoch} \]

Variance Reduction update:

\[ x^k_{m+1} = x^k_{m+1} - \eta(\nabla f_k(x^k_{m+1}) - \text{error}^k_{m+1}) \]
Suppose sequential ordering over data indices

\[ x_k^m : k\text{-th iterate of } m\text{-th epoch} \]

Variance Reduction update:

\[
x_{m+1}^{k+1} = x_{m+1}^k - \eta(\nabla f_k(x_{m+1}^k) - \text{error}_{m+1}^k)
\]

How to estimate the error correction term?
Suppose sequential ordering over data indices $x^k_m$:

$k$-th iterate of $m$-th epoch $x^k_m$

Variance Reduction update:

$$x^{k+1}_{m+1} = x^k_{m+1} - \eta (\nabla f_k(x^k_{m+1}) - \text{error}^k_{m+1})$$

How to estimate the error correction term?

Maintain a **table** of previous gradients
CENTRALVR ALGORITHM

Maintain a **table** of previous gradients
CENTRALVR ALGORITHM

Maintain a table of previous gradients

At end of $m$-th epoch:

\[
\begin{array}{c}
\nabla f_1(x_m^1) \\
\nabla f_2(x_m^2) \\
\vdots \\
\nabla f_{n-1}(x_m^{n-1}) \\
\n\nabla f_n(x_m^n)
\end{array}
\]
CENTRAL VR ALGORITHM

Maintain a table of previous gradients

At end of $m$-th epoch:
Average over stored gradients

\[
\bar{g}_m = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x^i_m)
\]

\[
\begin{align*}
\nabla f_1(x^1_m) \\
\nabla f_2(x^2_m) \\
\vdots \\
\nabla f_{n-1}(x^{n-1}_m) \\
\nabla f_n(x^n_m)
\end{align*}
\]
**CENTRAL VR ALGORITHM**

Maintain a **table** of previous gradients

At end of $m$-th epoch:

Average over stored gradients

$$
\bar{g}_m = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x^i_m)
$$

During $m+1$-th epoch

$$
\text{error}^k_{m+1} = \nabla f_k(x^k_m) - \bar{g}_m
$$

<table>
<thead>
<tr>
<th>$\nabla f_1(x^1_m)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nabla f_2(x^2_m)$</td>
</tr>
<tr>
<td>\vdots</td>
</tr>
<tr>
<td>$\nabla f_{n-1}(x^{n-1}_m)$</td>
</tr>
<tr>
<td>$\nabla f_n(x^n_m)$</td>
</tr>
</tbody>
</table>
CENTRALVR ALGORITHM

Maintain a **table** of previous gradients

At end of $m$-th epoch:
Average over stored gradients

$$
\overline{g}_m = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x^i_m)
$$

During $m+1$-th epoch

$$
\text{error}^k_{m+1} = \nabla f_k(x^k_m) - \overline{g}_m
$$

On $k$-th iteration:
- use stored gradient for $\text{error}^k_{m+1}$
- replace with new gradient $\nabla f_k(x^k_{m+1})$
CENTRAL VR ALGORITHM

Maintain a **table** of previous gradients

At end of $m$-th epoch:
Average over stored gradients

$$
\bar{g}_m = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x^i_m)
$$

During $m+1$-th epoch

$$
\text{error}_{m+1}^k = \nabla f_k(x^k_m) - \bar{g}_m
$$

On $k$-th iteration:

- use stored gradient for $\text{error}_{m+1}^k$
- replace with new gradient $\nabla f_k(x^k_{m+1})$

At end of $m+1$-th epoch: recalculate gradient average
PROPERTIES
PROPERTIES

• **Linear** convergence rate under strong convexity and Lipschitz smoothness (Theorem 1)
PROPERTIES

• **Linear** convergence rate under strong convexity and Lipschitz smoothness (Theorem 1)

• One gradient evaluation/iteration; \( n \) stored gradients
PROPERTIES

• **Linear** convergence rate under strong convexity and Lipschitz smoothness (Theorem 1)

• One gradient evaluation/iteration; $n$ stored gradients

• For simple models (linear or logistic regression): storing a gradient $\leftrightarrow$ storing a **scalar** $a^T x$
PROPERTIES

• **Linear** convergence rate under strong convexity and Lipschitz smoothness (Theorem 1)

• One gradient evaluation/iteration; \( n \) stored gradients

• For simple models (linear or logistic regression): storing a gradient \( \leftrightarrow \) storing a **scalar** \( a^T x \)

• Same algorithm with random permutations
DISTRIBUTED SETTING

\[
\min \sum_{i=1}^{n} f_i(x)
\]

example: least squares

\[
\min \frac{1}{2} \|Ax - b\|^2
\]

\[
\min \sum_{p=1}^{P} \frac{1}{2} \|A_p x - b_p\|^2
\]

data distributed over \( P \) local machines
**Centralized**: local nodes communicate only with central server

Want an algorithm with low communication requirements that can scale
DISTRIBUTED SETTING

**Centralized**: local nodes communicate only with central server

Want an algorithm with low communication requirements that can scale

Distributed CentralVR:
- local machines solve local problems
- central server aggregates solutions to solve global problem
SYNCHRONOUS CENTRALVR
SYNCHRONOUS CENTRALVR

- Local nodes receive current iterate and average gradient from central server (initially all 0s)
**SYNCHRONOUS CENTRALVR**

- Local nodes receive current iterate and average gradient from central server (initially all 0s)
- Each local node maintains **local table** of stored gradients
- Each local node runs one epoch of CentralVR

one epoch of CentralVR
SYNCHRONOUS CENTRALVR

• Local nodes receive current iterate and average gradient from central server (initially all 0s)
• Each local node stores local table of stored gradients
• Each local node runs one epoch of CentralVR
• Send current local iterate and local average gradient (averaged over local table) to central server
SYNCHRONOUS CENTRALVR

- Local nodes receive current iterate and average gradient from central server (initially all 0s)
- Each local node stores local table of stored gradients
- Each local node runs one epoch of CentralVR
- Send current local iterate and local average gradient (averaged over local table) to central server
SYNCHRONOUS CENTRALVR

- Local nodes receive current iterate and average gradient from central server (initially all 0s)
- Each local node stores **local table** of stored gradients
- Each local node runs one epoch of CentralVR
- Send current local iterate and local average gradient (averaged over local table) to central server
- Central server averages over the received parameters and sends them back to the local nodes
SYNCHRONOUS CENTRALVR

- Local nodes receive current iterate and average gradient from central server (initially all 0s)
- Each local node stores **local table** of stored gradients
- Each local node runs one epoch of CentralVR
- Send current local iterate and local average gradient (averaged over local table) to central server
- Central server averages over the received parameters and sends them back to the local nodes
ASYNCHRONOUS VERSION

Synchronous CentralVR can be easily extended to the Asynchronous case

Key Difference:
Local node sends back change in variables

\[ \Delta x_p^m = x_p^m - x_p^{m-1}, \quad \Delta \bar{g}_p^m = \bar{g}_p^m - \bar{g}_p^{m-1}, \]

Server integrates the change into the central average

\[ x = x + \frac{1}{p} x_s, \quad \bar{g} = \bar{g} + \frac{1}{p} \bar{g}_s \]

Previous contribution of node \( p \) replaced:
Faster working nodes don’t bias the solution
PROPERTIES
PROPERTIES

- One communication round/epoch (low)
PROPERTIES

• One communication round/epoch (**low**)

• Each node uses a **global** average gradient
PROPERTIES

• One communication round/epoch (low)

• Each node uses a global average gradient

• Global average gradient: helps keep local solution aligned with global solution
PROPERTIES

• One communication round/epoch (low)

• Each node uses a **global** average gradient

• Global average gradient: helps keep local solution **aligned** with global solution

• Asynchronous algorithm: **robust** to nodes working at drastically different speeds
EMPIRICAL RESULTS

Model: Ridge Regression

Datasets:
MILLIONSONG for regression: 463,715 samples
Toy data (random $A, b = Ax + \epsilon$): 5000 samples/node

Compared with:
• EASGD (Zhang, Choromanska, Lecun, 15)
• Asynchronous SVRG (Reddi et al, 15)
• Distributed SAGA (in CentralVR paper)
• Distributed SVRG (in CentralVR paper)

Check paper for additional experiments
Toy data set size increases linearly with number of workers
Maximum toy data set size: $5000 \times 960 = 4,800,000$
TAKEAWAYS

CentralVR

Leverages Variance Reduction techniques to speed up SGD

• Can efficiently scale up to hundreds of distributed computing nodes

• Low communication costs suitable for large-scale heterogenous distributed environments
THANKS!

Feel free to get in touch!

Email: sohamde@cs.umd.edu

Website: https://cs.umd.edu/~sohamde/