Gradient Coding:
Mitigating Stragglers in Distributed Gradient Descent.

Alex Dimakis

ejoint work with
R. Tandon, Q. Lei, UT Austin
N. Karampatziakis, Microsoft
Setup

• Problem: Large-scale learning.

• Given a lot of labeled examples and we want to fit a model, i.e. minimize a function.

• Modern models are becoming very complex.
• Training takes a very long time (dozens of hours, days, weeks)

Typical examples:
1. Big logistic regression (Ad prediction)
2. Training a deep neural network model. (vision, speech, NLP problems etc)
Gradient descent

Loss: \( L(\beta) = \sum_{i=1}^{d} \ell(\beta; x_i, y_i) \)

Gradient descent update step: \( \beta^{t+1} = \beta^t - \eta g^t \)

Gradient has the form: \( g^t = \sum_{i=1}^{d} \nabla \ell(\beta^t; (x_i, y_i)) \)
Gradient descent

Loss: $L(\beta) = \sum_{i=1}^{d} \ell(\beta; x_i, y_i)$

Gradient descent update step: $\beta^{t+1} = \beta^t - \eta g^t$

Gradient has the form: $g^t = \sum_{i=1}^{d} \nabla \ell(\beta^t; (x_i, y_i))$
Gradient descent

Loss: $L(\beta) = \sum_{i=1}^{d} \ell(\beta; x_i, y_i)$

Gradient descent update step: $\beta^{t+1} = \beta^t - \eta g^t$

Gradient has the form: $g^t = \sum_{i=1}^{d} \nabla \ell(\beta^t; (x_i, y_i))$
Distributed gradient descent

worker 1

model $\beta$

$D_1$

$D_2$

$D_3$

worker 2

model $\beta$

$D_4$

$D_5$

$D_6$

worker 3

model $\beta$

$D_7$

$D_8$

$D_9$

master
Distributed gradient descent

model \( \beta \)

worker 1

\( D_1 \)
\( D_2 \)
\( D_3 \)

\( g_1 \)

model \( \beta \)

worker 2

\( D_4 \)
\( D_5 \)
\( D_6 \)

model \( \beta \)

worker 3

\( D_7 \)
\( D_8 \)
\( D_9 \)

master
Distributed gradient descent
Distributed gradient descent

![Diagram of distributed gradient descent]

Worker 1:
- D_1
- D_2
- D_3

Worker 2:
- D_4
- D_5
- D_6

Worker 3:
- D_7
- D_8
- D_9

Model β

Master:

- Add gradients and update model

Worker 1:
- g_1

Worker 2:
- g_2

Worker 3:
- g_3

Model β
Distributed gradient descent

worker 1

\[ D_1 \]
\[ D_2 \]
\[ D_3 \]

\[ g_1 \]

worker 2

\[ D_4 \]
\[ D_5 \]
\[ D_6 \]

\[ g_2 \]

worker 3

\[ D_7 \]
\[ D_8 \]
\[ D_9 \]

\[ g_3 \]

master

add gradients and update model

model \( \beta \)
Distributed gradient descent

worker 1

D_1
D_2
D_3

worker 2

D_4
D_5
D_6

worker 3

D_7
D_8
D_9

worker 1 sends gradients g_1 to the master.
worker 2 sends gradients g_2 to the master.
worker 3 sends gradients g_3 to the master.

The master adds the gradients and updates the model β.

model β
Distributed Synchronous Gradient Descent

- All workers have the model and process different data examples to compute different gradients.
- We simply want to add them all up and update the model.
- After we update the model, send it back to the workers to re-compute gradients.
- Repeat this for a number of iterations.

- Problem: **Stragglers**. Some machines take too long to send their gradient.

- **Communication** seems to be the bottleneck.
How much Straggling is there?

50 t2-micro workers/ c38xlarge master/ 500k model

Avg. Time (in seconds)

Index of Worker
Gradient coding

model $\beta$

worker 1

$D_1$

$D_2$

$g_1$

$g_2$

worker 2

$D_2$

$D_3$

$g_2$

$g_3$

worker 3

$D_3$

$D_1$

$g_3$

$g_1$

add gradients and update model

master

model $\beta$
1. Add redundancy in computation. Gradients of data computed twice.

2. If you knew who is the straggler you would only communicate the needed gradients.
Gradient coding

worker 1
- $D_1$
- $D_2$
- $g_1$
- $g_2$

worker 2
- $D_2$
- $D_3$
- $g_2$
- $g_3$

worker 3
- $D_3$
- $D_1$
- $g_3$
- $g_1$

add gradients and update model

master

model $\beta$
Gradient coding

Worker 1: 
- \( D_1 \)
- \( D_2 \)
- \( g_1 \)
- \( g_2 \)
- \( g_1 + g_2 \)

Worker 2: 
- \( D_2 \)
- \( D_3 \)
- \( g_2 \)
- \( g_3 \)

Worker 3: 
- \( D_3 \)
- \( D_1 \)
- \( g_3 \)
- \( g_1 \)

Add gradients and update model master.
1. If you **knew** who is the straggler you can communicate only two vectors.

2. For any straggler: retrieve the sum from **any two**?
Gradient coding

worker 1

$D_1$

$D_2$

$0.5 \times g_1 + g_2$

worker 2

$D_2$

$D_3$

$g_2 - g_3$

worker 3

$D_3$

$D_1$

$0.5 \times g_1 + g_3$

add gradients and update model

master

model $\beta$
Gradient coding idea

- Design a matrix so that any $n$-s rows contain $[1, 1, 1]$ in their span.

$$B = \begin{pmatrix}
\frac{1}{2} & 1 & 0 \\
0 & 1 & -1 \\
\frac{1}{2} & 0 & 1 \\
\end{pmatrix}$$

- Trivial if $B$ is all ones. Want to minimize the amount of work per machine, i.e. the number of non-zeros per row of $B$. 
• Design a matrix so that any \( n-s \) rows contain \([1,1,1]\) in their span.

\[
B = \begin{pmatrix}
\frac{1}{2} & 1 & 0 \\
0 & 1 & -1 \\
\frac{1}{2} & 0 & 1
\end{pmatrix}
\]

• Trivial if \( B \) is all ones. Want to minimize the amount of work per machine, i.e. the number of non-zeros per row of \( B \).

**Theorem 1 (Lower Bound on \( B \)'s density).** Consider any scheme \((A, B)\) robust to any \( s \) stragglers, given \( n \) workers (with \( s < n \)) and \( k \) partitions. Then, if all rows of \( B \) have the same number of non-zeros, we must have: \( \|b_i\|_0 \geq \frac{k}{n}(s + 1) \) for any \( i \in [n] \).

set \( k=n \). Each data row must be processed \( s+1 \) times to tolerate any \( s \) stragglers.
• If $n$ divides $(s+1)$ very simple replication scheme is optimal.

• We have a scheme: Cyclic repetition that is optimal for any $n, s$.

• Designing codes for partial stragglers: One machine will only compute half of the gradients.
Experiments

- Real ec2 deployment, artificial straggling, n=12 machines.
Real contender: Ignore stragglers

• Just update model when 90% of machines have reported gradients. Used widely in practice.
Conclusions and outlook

- Accelerating distributed learning is a real problem.

- Coding techniques can be used in a few different ways
  - K. Lee, M. Lam, R. Pedarsani, D.S. Papailiopoulos, and K. Ramchandran
  - S. Dutta, V. Cadambe, P. Grover
  - S. Li, M. Maddah-Ali, S. Avestimehr

- Gradient coding can be used directly on any model (e.g. nonlinear). Only codes over the gradient vectors.

- Significant work ahead: Understanding bottlenecks for cost/bottleneck on AWS for different instances. Test over Tensorflow and MPI

- Partial Stragglers.

- Approximately containing ones vector in span.

- Better communication schemes (e.g. Trees).