Distributed machine learning using codes

Ardhendu Tripathy

Referred paper: Speeding up Distributed Machine Learning using Codes
Two important subroutines

• Matrix multiplication

• Stochastic gradient descent (below from Wikipedia)

\[ Q(w) = \frac{1}{n} \sum_{i=1}^{n} Q_i(w) \]
\[ w := w - \eta \nabla Q(w) = w - \eta \sum_{i=1}^{n} \nabla Q_i(w) \]

- Choose an initial vector of parameters \( w \) and learning rate \( \eta \).

- Repeat until an approximate minimum is obtained:
  - Randomly shuffle examples in the training set.
  - For \( i = 1, 2, \ldots, n \), do:
    - \( w := w - \eta \nabla Q_i(w) \).

Let's suppose we want to fit a straight line \( y = w_1 + w_2 x \) to a training set of two-dimensional points \( (x_1, y_1), \ldots, (x_n, y_n) \) using least squares. The objective function to be minimized is:

\[ Q(w) = \sum_{i=1}^{n} Q_i(w) = \sum_{i=1}^{n} (w_1 + w_2 x_i - y_i)^2. \]

The last line in the above pseudocode for this specific problem will become:

\[
\begin{bmatrix}
  w_1 \\
  w_2
\end{bmatrix}
:=
\begin{bmatrix}
  w_1 \\
  w_2
\end{bmatrix}
- \eta
\begin{bmatrix}
  2(w_1 + w_2 x_i - y_i) \\
  2x_i(w_1 + w_2 x_i - y_i)
\end{bmatrix}.
\]
Model for distributed computation
Mapreduce example – Word counting

- Count number of times three words A, B, C occur in six chapters
- Assign two different chapters to each worker, who counts number of times A, B, C occur
- Workers share their results
- Workers find total number of times word repeated
Effect of stragglers

Slowest worker decides execution time

Introduce redundancy to avoid slow worker?
Coded computation – Matrix multiplication

Baseline (uncoded), repetition-coded, Maximum Distance Separable-coded
Singleton bound

• A code $C$ is a collection of $n$-vectors with components from $q$ possible choices

• Hamming distance between two vectors is the number of places they differ in. Distance $d$ of $C$ is minimum hamming distance between any two codewords

• Removing the same $d-1$ symbols from each codeword still keeps them distinct. If $C$ is linear, i.e., $C$ is in a vector space and $c = x [l_k \ P]$

• $q^k \leq q^{(n-d+1)}$ MDS codes are those that achieve equality, they have the maximum error-correcting capability. From any $k$ components of $c$ one can recover the original $x$
Runtime of distributed algorithms

• Suppose n workers, runtime for ith worker is $T^i$, $T(j)$ is the jth smallest value in $\{T^i : i = 1, 2, \ldots, n\}$.

$$T_{\text{overall}}^{\text{uncoded}} = T(n) \overset{\text{def}}{=} \max\{T^1, T^2, \ldots, T^n\} \quad T_{\text{overall}}^{\text{coded}} = T(I) = \min_{i \in I} \max_{j \in i} T^j$$

• For repetition-coded:

$$\mathcal{I} = \left\{1, 2, \ldots, \frac{n}{k}\right\} \times \left\{\frac{n}{k} + 1, \frac{n}{k} + 2, \ldots, \frac{2n}{k}\right\} \times \cdots \times \left\{\frac{(k - 1)n}{k} + 1, \ldots, n\right\}$$

$$T_{\text{overall}}^{\text{Repetition-coded}} = \max_{i \in [k]} \min_{j \in [\frac{n}{k}]} \{T^{(i-1)\frac{n}{k} + j}\}$$

• For MDS-coded:

$$\mathcal{I} = \{i | i \in [n], |i| = k\}$$

$$T_{\text{overall}}^{\text{MDS-coded}} = T(k)$$
Probabilistic runtime model - exponential

\[ \Pr(T_0 \leq t) = 1 - e^{-\mu(t-1)}, \quad \forall t \geq 1 \]

- Runtime distribution of uncoded algorithm:
  \[ [F(nt)]^n \quad \mathbb{E}[T_{\text{overall}}^{\text{uncoded}}] = \frac{1}{n} \left( 1 + \frac{1}{\mu} \log n \right) \]

- Repetition-coded:
  \[ \left[ 1 - [1 - F(kt)]^{\frac{n}{k}} \right]^k \quad \mathbb{E}[T_{\text{overall}}^{\text{Repetition-coded}}] = \frac{1}{k} \left( 1 + \frac{k}{n\mu} \log k \right) \]

- MDS-coded: kth order-statistic
  \[ \mathbb{E}[T_{\text{overall}}^{\text{MDS-coded}}] = \frac{1}{k} \left( 1 + \frac{1}{\mu} \log \left( \frac{n}{n-k} \right) \right) \]
Coded shuffling – stochastic optimization

- Gain when cost of broadcasting to all workers is less than unicasting same message to each worker
- True for wireless setting, even for wireline setting. E.g. Message Passing Interface (MPI) has routine MPI_Bcast where the ratio of costs grows as $\Theta(n/ \log n)$. 
Coded vs. Uncoded shuffling

• Entire matrix A has q rows, each worker’s cache can store s, n workers
• For ith worker, cache content is $C_i$, new data required is $S_i$. Under uniform random permutation, w.h.p. as q gets large

$$|C_i \cap S_i| = \frac{q}{n}(1 - s/q)$$

$$R_u = n \times \frac{q}{n}(1 - s/q) = q(1 - s/q)$$
Coded vs. Uncoded shuffling

- Exclusive cache content
  \[ \tilde{C}_\mathcal{I} = \bigcap_{i \in \mathcal{I}} C_i \cap \bigcap_{i' \in [n] \setminus \mathcal{I}} \tilde{C}_{i'} \]
- Example: \( n=3 \)

\[ M_1 = A(S_1 \cap \tilde{C}_2) + A(S_2 \cap \tilde{C}_1) \]
\[ M_2 = A(S_1 \cap \tilde{C}_3) + A(S_3 \cap \tilde{C}_1) \]
\[ M_3 = A(S_2 \cap \tilde{C}_3) + A(S_3 \cap \tilde{C}_2) \]
\[ M_4 = A(S_1 \cap \tilde{C}_{2,3}) + A(S_2 \cap \tilde{C}_{1,3}) + A(S_3 \cap \tilde{C}_{1,2}) \]