Dimension Independent Matrix Square using MapReduce (DIMSUM)

Reza Zadeh
Institute for Computational and Mathematical Engineering

Stanford

Introduction

- Given $m \times n$ matrix A with entries in $[0,1]$ and $m \gg n$, compute $A^T A$.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

- A is tall and skinny, example values $m = 10^{12}$, $n = 10^6$.
- A has sparse *rows*, each row has at most L nonzeros.
- A is stored across thousands of machines and cannot be streamed through a single machine.

Guarantees

- Preserve singular values of $A^T A$ with ϵ relative error paying shuffle size $O(n^2/\epsilon^2)$ and reduce-key complexity $O(n/\epsilon^2)$. i.e. independent of m.
- Preserve specific entries of $A^T A$, then we can reduce the shuffle size to $O(n \log(n)/s)$ and reduce-key complexity to $O(\log(n)/s)$ where s is the minimum similarity for the entries being estimated. Similarity can be via Cosine, Dice, Overlap, or Jaccard.

Computing All Pairs of Dot Products

- We have to find dot products between all pairs of columns of A
- We prove results for general matrices, but can do better for those entries with $\cos(i,j) \geq s$
- Cosine similarity: a widely used definition for "similarity" between two vectors

$$\cos(i,j) = \frac{c_i^T c_j}{||c_i|| \, ||c_j||}$$

- c_i is the $i'th$ column of A

MapReduce

- Input gets dished out to the mappers roughly equally. Two performance measures
- 1) Shuffle size: shuffling the data output by the mappers to the correct reducer is expensive
- 2) Largest reduce-key: can't send too much of the data to a single reducer

Naive Implementation

Algorithm 1 NaiveMapper(r_i)

for all pairs (a_{ij}, a_{ik}) in r_i **do**
  Emit $((c_j, c_k) \rightarrow a_{ij} a_{ik})$
end for

Algorithm 2 NaiveReducer($(c_i, c_j), \langle v_1, \ldots, v_R \rangle$)

output $c_i^T c_j \rightarrow \sum_{i=1}^{R} v_i$

- Shuffle size: $O(mL^2)$ and largest reduce-key: $O(m)$
- Both depend on m, the larger dimension, and are intractable for $m = 10^{12}$, $L = 100$.
- We'll bring both down via clever sampling

DIMSUM

Algorithm 3 DIMSUMMapper(r_i)

for all pairs (a_{ij}, a_{ik}) in r_i **do**
  With probability $\min\left(1, \gamma \frac{1}{||c_j|| \, ||c_k||}\right)$
  emit $((c_j, c_k) \rightarrow a_{ij} a_{ik})$
end for

Algorithm 4 DIMSUMReducer($(c_i, c_j), \langle v_1, \ldots, v_R \rangle$)

if $\frac{\gamma}{||c_i|| \, ||c_j||} > 1$ **then**
  output $b_{ij} \rightarrow \frac{1}{||c_i|| \, ||c_j||} \sum_{i=1}^{R} v_i$
else
  output $b_{ij} \rightarrow \frac{1}{\gamma} \sum_{i=1}^{R} v_i$
end if

Analysis for DIMSUM

Four things to prove:

- Shuffle size: $O(nL\gamma)$
- Largest reduce-key: $O(\gamma)$
- The sampling scheme preserves similarities when $\gamma = \Omega(\log(n)/s)$
- The sampling scheme preserves singular values when $\gamma = \Omega(n/\epsilon^2)$

Shuffle Size and Largest Reduce Key

- Let H be the smallest nonzero entry in magnitude, after all entries of A have been scaled to be in $[0,1]$
- E.g. for $\{0,1\}$ matrices, we have $H = 1$
- Shuffle size is bounded by $O(nL\gamma/H^2)$
- Largest reduce-key is bounded by $O(\gamma)$

Correctness

- Since higher magnitude columns are sampled with lower probability, are we guaranteed to obtain correct results w.h.p.?
- Yes. By setting γ correctly.
- Preserve similarities when $\gamma = \Omega(\log(n)/s)$
- Preserve singular values when $\gamma = \Omega(n/\epsilon^2)$

Theorem

Let A be an $m \times n$ tall and skinny ($m > n$) matrix. If $\gamma = \Omega(n/\epsilon^2)$ and D a diagonal matrix with entries $d_{ii} = ||c_i||$, then the matrix B output by DIMSUM satisfies,

$$\frac{||DBD - A^T A||_2}{||A^T A||_2} \leq \epsilon$$

with probability at least $1/2$.

Theorem

For any two columns c_i and c_j having $\cos(c_i, c_j) \geq s$, let B be the output of DIMSUM with entries $b_{ij} = \frac{1}{\gamma} \sum_{k=1}^{m} X_{ijk}$ with X_{ijk} as the indicator for the k'th coin in the call to DIMSUMMapper. Now if $\gamma = \Omega(\alpha/s)$, then we have,

$$\Pr\left[||c_i|| \, ||c_j|| b_{ij} > (1+\delta)[A^T A]_{ij} \right] \leq \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}} \right)^\alpha$$

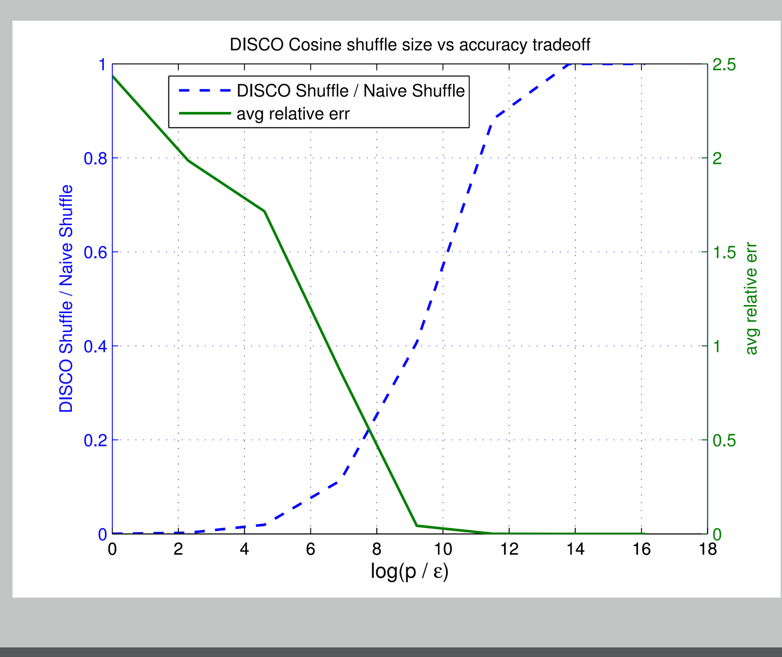
and

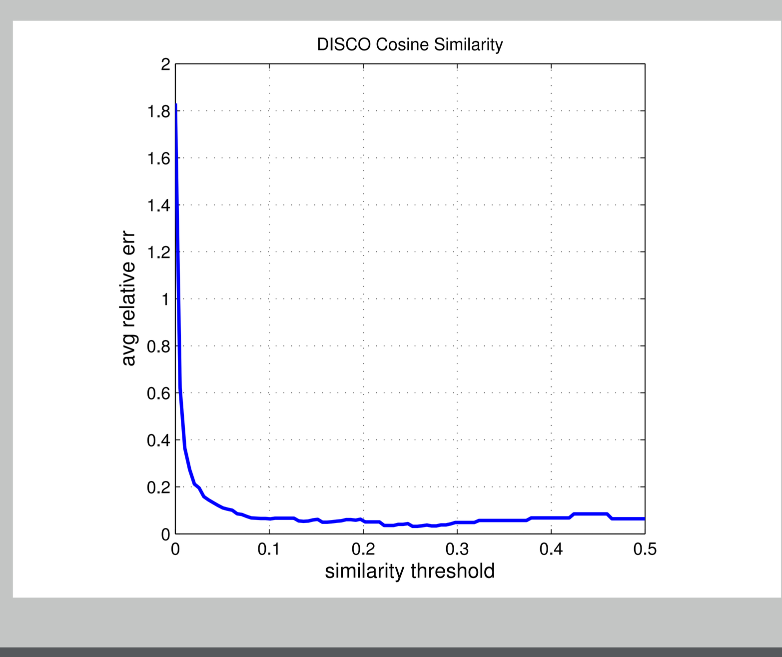
$$\Pr\left[||c_i|| \, ||c_j|| b_{i,j} < (1-\delta)[A^T A]_{ij} \right] < \exp(-\alpha\delta^2/2)$$

Live Applications

- Large scale live at `twitter.com`

Experiments





http://reza-zadeh.com
rezab@stanford.edu