Dimension Independent Similarity Computation

Reza Bosagh Zadeh
Joint work with Ashish Goel

ICME
INSTITUTE for COMPUTATIONAL & MATHEMATICAL ENGINEERING at STANFORD UNIVERSITY

ICME Seminar February 2013

Outline

Computing $A^T A$

- Given $N \times D$ matrix A with $\{0, 1\}$ entries and $N \gg D$, compute $A^T A$.

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

- A is tall and skinny, example values $N = 10^{12}$, $D = 10^6$.
- A has sparse *rows*, each row has at most L nonzeros.
- A is stored across thousands of machines.

Computing All Pairs of Cosine Similarities

- We can focus on computing cosine similarities between pairs of columns of A

$$\cos(i,j) = \frac{\#(w_i, w_j)}{\sqrt{\#(w_i)}\sqrt{\#(w_j)}}$$

- w_i is the $i'th$ column of A
- Since A has 0-1 entries, $\#(w_i, w_j) = w_i^T w_j$ and $\sqrt{\#(w_i)} = ||w_i||_2$
- We focus on provable results for large entries, in particular those with $\cos(i,j) \geq \epsilon$

Ubiquitous problem

MapReduce

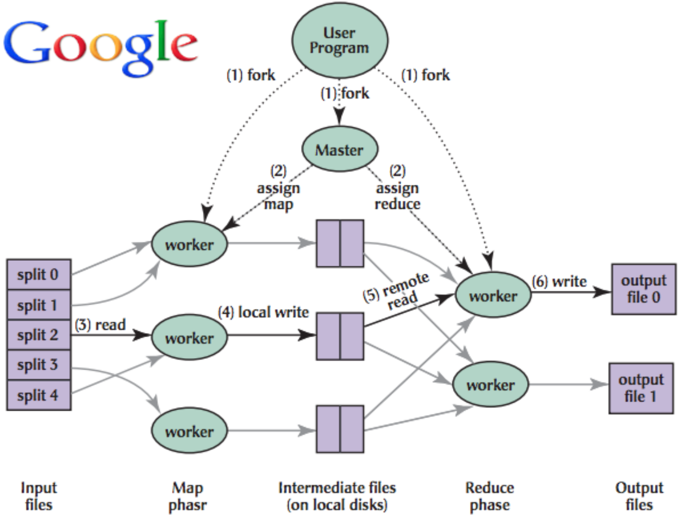
- With such large datasets (e.g. $N = 10^{12}$), we must use many machines.
- Biggest clusters of computers use MapReduce
- MapReduce is the tool of choice in such distributed systems
- With so many machines (around 1000), CPU power is abundant, but communication is expensive
- 2 Minute description of MapReduce...

MapReduce


```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

MapReduce


Google
User Program
(1) fork
(1) fork
(1) fork
Master
(2) assign map
(2) assign reduce
worker
worker
worker
worker
worker
split 0
split 1
split 2
split 3
split 4
(3) read
(4) local write
(5) remote read
(6) write
output file 0
output file 1
Input files
Map phasr
Intermediate files (on local disks)
Reduce phase
Output files

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
- First pass at implementing $\cos(i, j)$ in MapReduce...

Naive Implementation

1. Given row t, Map with NaiveMapper (Algorithm 1)
2. Reduce using the NaiveReducer (Algorithm 2)

Algorithm 1 NaiveMapper(t)

> **for** all pairs (w_1, w_2) in t **do**
>   emit $((w_1, w_2) \to 1)$
> **end for**

Algorithm 2 NaiveReducer($(w_1, w_2), \langle r_1, \ldots, r_R \rangle$)

> $a = \sum_{i=1}^{R} r_i$
> output $\frac{a}{\sqrt{\#(w_1)\#(w_2)}}$

Analysis for First Pass

- Very easy analysis
- 1) Shuffle size: $O(NL^2)$
- 2) Largest reduce-key: $O(N)$
- Both depend on N, the dimension, and are intractable for $N = 10^{12}, L = 100$.
- We'll bring both down via clever sampling

DISCO Algorithm

Algorithm 3 DISCOMapper(t)

for all pairs (w_1, w_2) in t **do**
With probability

$$\frac{p}{\epsilon}\frac{1}{\sqrt{\#(w_1)}\sqrt{\#(w_2)}}$$

emit $((w_1, w_2) \to 1)$
end for

Algorithm 4 DISCOReducer($(w_1, w_2), \langle r_1, \ldots, r_R \rangle$)

$a = \sum_{i=1}^{R} r_i$
output $a\frac{\epsilon}{p}$

Analysis for DISCO

Three things to prove:

1. Shuffle size: $O(DL\log(D)/\epsilon)$
2. Largest reduce-key: $O(\log(D)/\epsilon)$
3. The sampling scheme actually works with high probability

Shuffle size for DISCO

Theorem

The expected shuffle size for DISCOMapper is $O(DL\log(D)/\epsilon)$.

Proof.

The expected contribution from each pair of words will constitute the shuffle size:

$$\sum_{i=1}^{D}\sum_{j=i+1}^{D}\sum_{k=1}^{\#(w_i,w_j)} \Pr[\text{CosineSampleEmit}(w_i, w_j)]$$

$$= \sum_{i=1}^{D}\sum_{j=i+1}^{D} \#(w_i, w_j)\Pr[\text{CosineSampleEmit}(w_i, w_j)]$$

Shuffle size for DISCO

Proof.

$$\leq \sum_{i=1}^{D} \sum_{j=i+1}^{D} \frac{p}{\epsilon} \frac{\#(w_i, w_j)}{\sqrt{\#(w_i)}\sqrt{\#(w_j)}}$$

Shuffle size for DISCO

Proof.

$$\leq \sum_{i=1}^{D} \sum_{j=i+1}^{D} \frac{p}{\epsilon} \frac{\#(w_i, w_j)}{\sqrt{\#(w_i)}\sqrt{\#(w_j)}}$$

$$\text{(by AM-GM)} \leq \frac{p}{2\epsilon} \sum_{i=1}^{D} \sum_{j=i+1}^{D} \#(w_i, w_j)\left(\frac{1}{\#(w_i)} + \frac{1}{\#(w_j)}\right)$$

Shuffle size for DISCO

Proof.

$$\leq \sum_{i=1}^{D} \sum_{j=i+1}^{D} \frac{p}{\epsilon} \frac{\#(w_i, w_j)}{\sqrt{\#(w_i)}\sqrt{\#(w_j)}}$$

(by AM-GM) $$\leq \frac{p}{2\epsilon} \sum_{i=1}^{D} \sum_{j=i+1}^{D} \#(w_i, w_j)\left(\frac{1}{\#(w_i)} + \frac{1}{\#(w_j)}\right)$$

$$\leq \frac{p}{\epsilon} \sum_{i=1}^{D} \frac{1}{\#(w_i)} \sum_{j=1}^{D} \#(w_i, w_j)$$

Shuffle size for DISCO

Proof.

$$\leq \sum_{i=1}^{D} \sum_{j=i+1}^{D} \frac{p}{\epsilon} \frac{\#(w_i, w_j)}{\sqrt{\#(w_i)}\sqrt{\#(w_j)}}$$

$$\text{(by AM-GM)} \quad \leq \frac{p}{2\epsilon} \sum_{i=1}^{D} \sum_{j=i+1}^{D} \#(w_i, w_j)\left(\frac{1}{\#(w_i)} + \frac{1}{\#(w_j)}\right)$$

$$\leq \frac{p}{\epsilon} \sum_{i=1}^{D} \frac{1}{\#(w_i)} \sum_{j=1}^{D} \#(w_i, w_j)$$

$$\leq \frac{p}{\epsilon} \sum_{i=1}^{D} \frac{1}{\#(w_i)} L\#(w_i) = \frac{p}{\epsilon} LD = O(DL\log(D)/\epsilon)$$

$\square$

Shuffle size for DISCO

- It is easy to see via Chernoff bounds that the above shuffle size is obtained with high probability.
- $O(DL\log(D)/\epsilon)$ has no dependence on the dimension N, this is the heart of DISCO.
- Happens because higher magnitude columns are sampled with lower probability:

$$\frac{p}{\epsilon}\frac{1}{\sqrt{\#(w_1)}\sqrt{\#(w_2)}}$$

Largest reduce key for DISCO

- Each reduce key receives at most $\frac{p}{\epsilon}$ values (the oversampling parameter)
- Immediately get that reduce-key complexity is $O(\log(D)/\epsilon)$
- Also independent of dimension N. Happens because high magnitude columns are sampled with lower probability.

Correctness

- Since higher magnitude columns are sampled with lower probability, are we guaranteed to obtain correct results w.h.p.?
- Yes. But provably only for points that have $\cos(i, j) \geq \epsilon$

Correctness

Theorem

For any two words x and y having $\cos(x, y) \geq \epsilon$*, let* $X_1, X_2, \ldots, X_{\#(x,y)}$ *represent indicators for the coin flip in calls to DISCOMapper with* x, y *parameters, and let* $X = \sum_{i=1}^{\#(x,y)} X_i$*. For any* $1 > \delta > 0$*, we have*

$$\Pr\left[\frac{\epsilon}{p}X > (1+\delta)\cos(x,y)\right] \leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{p}$$

and

$$\Pr\left[\frac{\epsilon}{p}X < (1-\delta)\cos(x,y)\right] < e^{-p\delta^2/2}$$

Relative error guaranteed to be low with high probability.

Correctness

Proof.

- In the paper at `http://reza-zadeh.com`
- Uses standard concentration inequality for sums of indicator random variables.
- Ends up requiring that the oversampling parameter p be set to $p = \log(D^2) = 2\log(D)$.

$\square$

Experiments

- Large scale experiment live at `twitter.com`

- Smaller scale experiment with points as words, and dimensions as tweets
- $N = 200M, D = 1000, L = 10$

Experiments

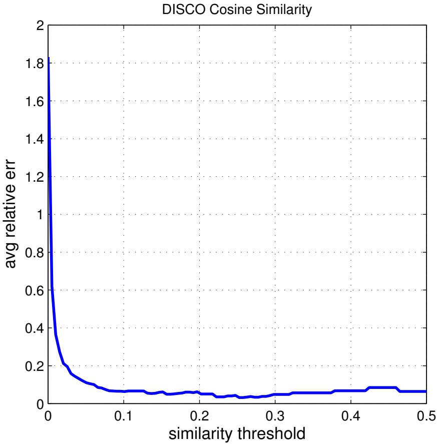


Figure: Average error for all pairs with similarity $\geq \epsilon$. DISCO estimated Cosine error decreases for more similar pairs.

Experiments

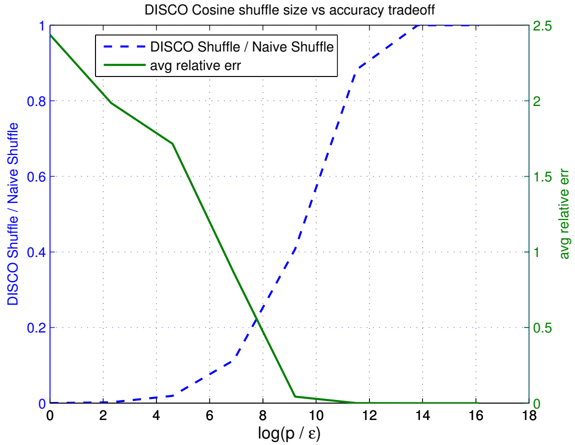


Figure: As p/ϵ increases, shuffle size increases and error decreases. There is no thresholding for highly similar pairs here.

Other Similarity Measures

This all works for many other similarity measures.

Similarity	Definition	Shuffle Size	Reduce-key size
Cosine	$\frac{\#(x,y)}{\sqrt{\#(x)}\sqrt{\#(y)}}$	$O(DL\log(D)/\epsilon)$	$O(\log(D)/\epsilon)$
Jaccard	$\frac{\#(x,y)}{\#(x)+\#(y)-\#(x,y)}$	$O((D/\epsilon)\log(D/\epsilon))$	$O(\log(D/\epsilon)/\epsilon)$
Overlap	$\frac{\#(x,y)}{\min(\#(x),\#(y))}$	$O(DL\log(D)/\epsilon)$	$O(\log(D)/\epsilon)$
Dice	$\frac{2\#(x,y)}{\#(x)+\#(y)}$	$O(DL\log(D)/\epsilon)$	$O(\log(D)/\epsilon)$

Table: All sizes are independent of N, the dimension. These are bounds for shuffle size without combining. Combining can only bring down these sizes.

Locality Sensitive Hashing

- MinHash from the Locality-Sensitive-Hashing family can have its vanilla implementation greatly improved by DISCO.
- Theorems for shuffle size and correctness in paper.

Fin.

- Consider DISCO if you ever need to compute $A^T A$ for large sparse A
- Many more experiments and results at `reza-zadeh.com`
- Thanks!