Dimension Independent Matrix Square using MapReduce

Reza Bosagh Zadeh
Institute for Computational and Mathematical Engineering
rezab@stanford.edu

Gunnar Carlsson
Mathematics Department
gunnar@math.stanford.edu

Abstract

We compute the singular values of an $m \times n$ tall and skinny ($m \gg n$) sparse matrix $A$ without dependence on $m$, for very large $m$. In particular, we give a simple nonadapative sampling scheme where the singular values of $A$ are estimated within relative error with high probability. Our proven bounds focus on the MapReduce framework, which has become the de facto tool for handling such large matrices that cannot be stored or even streamed through a single machine.

On the way, we give a general method to compute $A^T A$. We preserve singular values of $A^T A$ with $\epsilon$ relative error with shuffle size $O(n^2/\epsilon^2)$ and reduce-key complexity $O(n/\epsilon^2)$. We further show that if only specific entries of $A^T A$ are required, 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 cosine similarity for the entries being estimated. All of our bounds are independent of $m$, the larger dimension.

1 Introduction

There has been a flurry of work to solve problems in numerical linear algebra via fast approximate randomized algorithms. Starting with [18] many algorithms have been proposed over older algorithms [12, 13, 14, 10, 15, 16, 11, 17, 23, 4, 5, 21], with results satisfying the traditional Monte Carlo performance guarantees: small error with high probability.

These proposed algorithms require either streaming, or having access to the entire matrix $A$ on a single machine, or communicating too much data between machines. This is not feasible for very large $m$ (for example $m = 10^{13}$). In such cases, $A$ cannot be stored or streamed through a single machine - let alone be used in computations. For such cases, MapReduce [8] has become the de facto tool for handling very large datasets.

MapReduce is a programming model for processing large data sets, typically used to do distributed computing on clusters of commodity computers. With large amount of processing power at hand, it is very tempting to solve problems by brute force. However, we combine clever sampling techniques with the power of MapReduce to extend its utility.

Given an $m \times n$ matrix $A$ with each row having at most $L$ nonzero entries, we show how to compute the singular values and and right singular vectors of $A$ without dependence on $m$, in a MapReduce environment. The SVD of $A$ is written $A = U \Sigma V^T$, where $U$ is $m \times n$, $\Sigma$ is $n \times n$, and $V$ is $n \times n$.

We compute $\Sigma$ and $V$. We do this by first computing $A^T A$, which we do without dependence on $m$. Since $A^T A = V \Sigma^2 V^T$ is $n \times n$, for small $n$ (for example $n = 10^4$) we can compute the eigen-decomposition of $A^T A$ directly and retrieve $V$ and $\Sigma$. What remains is to compute $A^T A$ efficiently and without harming its singular values, which is what the rest of the paper is focused on.

Our main result is Algorithms 3 and 4, along with proven guarantees given in Theorem 4.2, which proves a relative error bound using the spectral norm. The proof uses a new singular value concentration inequality from Latala [22] that has not seen much usage by the theoretical computer science community.
2 Formal Preliminaries

Label the columns of $A$ as $c_1, \ldots, c_n$, rows as $r_1, \ldots, r_m$, and the individual entries as $a_{ij}$. The matrix is stored row-by-row on disk and read via mappers. We focus on the case where each dimension is sparse with at most $L$ nonzeros per row therefore the natural way to store the data is to segment into rows.

We use the matrix spectral norm throughout, which for any $m \times n$ matrix $A$ is defined as

$$||A||_2 = \max_{x \in \mathbb{R}^m, y \in \mathbb{R}^n} \frac{x^T Ay}{||x||_2 ||y||_2}$$

Unless otherwise denoted, the norm used anywhere in this paper is the spectral norm, which for regular vectors degenerates to the vector $l_2$ norm.

We concentrate on the regime where $m$ is very large, e.g. $m = 10^{13}$, but $n$ is not too large, e.g. $n = 10^4$, such that we can compute the SVD of an $n \times n$ dense matrix on a single machine. The magnitudes of each column is assumed to be loaded into memory and available to both the mappers and reducers. The magnitudes of each column are natural values to have computed already, or can be computed with a trivial mapreduce.

2.1 Naive Computation

The naive way to compute $A^T A$ on MapReduce is to materialize all dot products between columns of $A$ trivially. For purposes of demonstrating the complexity measures for MapReduce, we briefly write down the Naive algorithm to compute $A^T A$.

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

2.2 Complexity Measures

There are two main complexity measures for MapReduce: “shuffle size”, and “reduce-key complexity”. These complexity measures together capture the bottlenecks when handling data on multiple machines: first we can’t have too much communication between machines, and second we can’t overload a single machine. The number of emissions in the map phase is called the “shuffle size”, since that data needs to be shuffled around the network to reach the correct reducer. The maximum number of items reduced to a single key is called the “reduce-key complexity” and measures how overloaded a single machine may become [19].

It can be easily seen that the naive approach for computing $A^T A$ will have $O(mL^2)$ emissions, which for the example parameters we gave ($m = 10^{13}$, $n = 10^4$, $L = 20$) is infeasible. Furthermore, the maximum number of items reduced to a single key is $m$. Thus the “reduce-key complexity” for the naive scheme is $m$.

We can drastically reduce the shuffle size and reduce-key complexity by some clever sampling with the DIMSUM scheme described in this paper. In this case, the output of the reducers are random variables whose expectations are cosine similarities i.e. normalized entries of $A^T A$. Two proofs are needed to justify the effectiveness of this scheme. First, that the expectations are indeed correct and obtained with high probability, and second, that the shuffle size is greatly reduced. We prove both of these claims. In particular, in addition to correctness, we prove that for relative error $\epsilon$, the shuffle size of our scheme is only $O(n^2/\epsilon^2)$, with no dependence on the dimension $m$, hence the title of this paper.

This means as long as there are enough mappers to read the data, our sampling scheme can be used to make the shuffle size tractable. Furthermore, each reduce-key gets at most $O(n/\epsilon^2)$ values, thus making the reduce-key complexity...
tractable, too. Within Twitter Inc, we use the DIMSUM sampling scheme to compute similar users \[27, 20\]. We have also used the scheme to find highly similar pairs of words, by taking each dimension to be the indicator vector that signals in which tweets the word appears. We empirically verified the proven claims in this paper, but do not report experimental results since we are primarily focused on the proofs.

2.3 Related Work

Frieze et al. \[18\] introduced a sampling procedure where rows and columns of \(A\) are picked with probabilities proportional to their squared lengths and used that to compute an approximation to \(A^T A\). Later \[1\] and \[2\] improved the sampling procedure. To implement these approximations to \(A^T A\) on MapReduce one would need a shuffle size dependent on \(m\) or overload a single machine. We improve this to be independent of \(m\) both in shuffle size and reduce-key complexity.

Later on \[9\] found an adaptive sampling scheme to improve the scheme of \[18\]. Since the scheme is adaptive, it would require too much communication between machines holding \(A\). In particular a MapReduce implementation would still have shuffle size dependent on \(m\), and require many (more than 1) iterations.

There has been some effort to reduce the number of passes required through the matrix \(A\) using little memory, in the streaming model \[6\]. The question was posed by \[23\] to determine in the streaming model various linear algebraic quantities. The problem was posed again by \[24\] who asked about the time and space required for an algorithm not using too many passes. The streaming model is a good one if all the data can be streamed through a single machine, but with \(m\) so large, it is not possible to stream \(A\) through a single machine. Splitting the work of reading \(A\) across many mappers is the job of the MapReduce implementation and one of its major advantages \[8\].

There has been recent work specifically targeted at computing the SVD on MapReduce \[3\] in a stable manner via QR factorizations and bypassing \(A^T A\), with shuffle size and reduce-key complexity both dependent on \(m\).

In addition to computing entries of \(A^T A\), our sampling scheme can be used to implement many similarity measures. We can use the scheme to efficiently compute four similarity measures: Cosine, Dice, Overlap, and the Jaccard similarity measures, with details and experiments given in \[26, 20\], whereas this paper is more theoretically focused.

3 Algorithm

Our algorithm to compute \(A^T A\) efficiently is given below in Algorithms 3 and 4.

**Algorithm 3** DIMSUMMapper\(r_i\)

```python
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\((c_i, c_j, \langle v_1, \ldots, v_R \rangle)\)

```python
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
```

It is important to observe what happens if the output ‘probability’ is greater than 1. We certainly Emit, but when the output probability is greater than 1, care must be taken while reducing to scale by the correct factor, since it won’t be
correct to divide by $\gamma$, which is the usual case when the output probability is less than 1. Instead, the sum in Algorithm 4 obtains the dot product, because for the pairs where the output probability is greater than 1, DIMSUMMapper effectively always emits. We do not repeat this point later in the paper, nonetheless it is an important one which arises during implementation.

4 Correctness

Before we move onto the correctness of the algorithm, we must state Latala’s Theorem [22]. This theorem talks about a general model of random matrices whose entries are independent centered random variables with some general distribution (not necessarily normal). The largest singular value (the spectral norm) can be estimated by Latala’s theorem for general random matrices with non-identically distributed entries:

**Theorem 4.1.** (Latala’s theorem [22]). Let $X$ be a random matrix whose entries $x_{ij}$ are independent centered random variables with finite fourth moment. Denoting $\|X\|_2$ as the matrix spectral norm, we have

$$
\mathbb{E}\|X\|_2 \leq C \left[ \max_i \left( \sum_j \mathbb{E} x_{ij}^2 \right)^{1/2} + \max_j \left( \sum_i \mathbb{E} x_{ij}^2 \right)^{1/2} + \left( \sum_{i,j} \mathbb{E} x_{ij}^4 \right)^{1/4} \right].
$$

We analyze the second and fourth central moments of the entries of the estimate for $A^T A$, and show that by Latala’s theorem, the singular values are preserved with high probability. Let the matrix output by the DIMSUM algorithm be called $B$ with entries $b_{ij}$. Notice that this is an $n \times n$ matrix of cosine similarities between columns of $A$. Define a diagonal matrix $D$ with $d_{ii} = ||c_i||$. Then we can undo the cosine similarity normalization to obtain an estimate for $A^T A$ by using $DBD$. This effectively uses the cosine similarities between columns of $A$ as an importance sampling scheme. We have the following theorem:

**Theorem 4.2.** 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 (Algorithms 3 and 4) satisfies,

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

with probability at least $1/2$.

**Proof.** We define the indicator variable $X_{ijk}$ to take value $a_{ki}a_{kj}$ with probability $p_{ij} = \frac{1}{||c_i|| ||c_j||}$ on the $k$'th call to DIMSUMMapper, and zero with probability $1 - p_{ij}$.

$$
X_{ijk} = \begin{cases} 
  a_{kj}a_{kj} & \text{with prob. } p_{ij} \\
  0 & \text{with prob. } 1 - p_{ij}
\end{cases}
$$

Then we can write the entries of $B$ as

$$
b_{ij} = \frac{1}{\gamma} \sum_{k=1}^m X_{ijk}
$$

Since we give relative error bounds and singular values scale trivially, we can assume $A$ has all entries in $[0, 1]$. i.e. any scaling of the input matrix will have the same relative error guarantee. This assumption will be useful because we first prove an absolute error bound, then use that to prove a relative error bound. It should be clear from the definitions that in expectation

$$
\mathbb{E}[B] = D^{-1}A^T A D^{-1} \quad \text{and} \quad \mathbb{E}[DBD] = A^T A
$$

With these definitions, we now move onto bounding $\mathbb{E}[||B - D^{-1}A^T A D^{-1}||]$. With the goal of invoking Latala’s theorem, we analyze $\mathbb{E}[(b_{ij} - Eb_{ij})^2]$ and $\mathbb{E}[(b_{ij} - Eb_{ij})^4]$.

Now define $\#(i, j)$ as the number of dimensions in which $c_i$ and $c_j$ are both nonzero, i.e. the number of $k$ for which $a_{ki}a_{kj}$ is nonzero, and further define $i \cap j$ as the set of indices for which $a_{ki}a_{kj}$ is nonzero.

Clearly, $\mathbb{E}[(b_{ij} - Eb_{ij})^2]$ is the variance of $b_{ij}$, which is the sum of $\#(i, j)$ weighted indicator random variables. Thus we have
\[ E[(b_{ij} - E b_{ij})^2] = \text{Var}[b_{ij}] = \frac{1}{\gamma^2} \sum_{k \in i'j} \text{Var}[X_{ijk}] \]

\[ = \frac{1}{\gamma^2} \sum_{k \in i'j} a_k^2 a_{kj}^2 p_{ij}(1 - p_{ij}) \]

\[ \leq \frac{1}{\gamma^2} \sum_{k \in i'j} a_k^2 a_{kj}^2 p_{ij} \]

\[ = \frac{1}{\gamma^2} \sum_{k \in i'j} a_k^2 a_{kj}^2 \gamma \frac{1}{||c_i|| ||c_j||} \]

Now by the Arithmetic-Mean Geometric-Mean inequality,

\[ \leq \frac{1}{2\gamma^2} \sum_{k \in i'j} a_k^2 a_{kj}^2 \gamma \left( \frac{1}{||c_i||^2} + \frac{1}{||c_j||^2} \right) \]

\[ = \frac{1}{2\gamma} \sum_{k \in i'j} a_k^2 a_{kj}^2 \left( \frac{1}{||c_i||^2} + \frac{1}{||c_j||^2} \right) \]

\[ \leq \frac{1}{\gamma} \sum_{k \in i'j} a_k^2 a_{kj}^2 \left( \frac{1}{||c_j||^2} \right) \]

\[ \leq \frac{1}{\gamma} \sum_{k \in i'j} a_k^2 a_{kj}^2 \leq \frac{1}{\gamma} \]

Thus we have \( E[(b_{ij} - E b_{ij})^2] \leq \frac{1}{\gamma} \). It remains to bound the fourth central moment of \( b_{ij} \). We use a counting trick to achieve this bound:

\[ E[(b_{ij} - E b_{ij})^4] = \frac{1}{\gamma^4} \mathbb{E} \left[ \left( \sum_{k \in i'j} X_{ijk} - a_k a_{kj} p_{ij} \right)^4 \right] \]

\[ = \frac{1}{\gamma^4} \mathbb{E} \left[ \sum_{q, r, s, t \in i'j} (X_{ijq} - a_q a_{jq} p_{ij})(X_{ijr} - a_r a_{jr} p_{ij})(X_{ij} - a_s a_{js} p_{ij})(X_{ijt} - a_t a_{jt} p_{ij}) \right] \]

\[ = \frac{1}{\gamma^4} \sum_{q, r, s, t \in i'j} \mathbb{E} [(X_{ijq} - a_q a_{jq} p_{ij})(X_{ijr} - a_r a_{jr} p_{ij})(X_{ij} - a_s a_{js} p_{ij})(X_{ijt} - a_t a_{jt} p_{ij})] \]

which effectively turns this into a counting problem. The terms in the sum on the last expression are 0 unless either \( q = r = s = t \), which happens \(#(i, j)#\times times, or there are two pairs of matching indices, which happens \( \binom{\#(i, j)}{2} \times \binom{\#(i, j)}{2} \) times. Continuing, this gives us

\[ = \frac{1}{\gamma^4} \sum_{k \in i'j} \mathbb{E} [(X_{ijk} - a_k a_{kj} p_{ij})^4] + \frac{1}{\gamma^4} \sum_{q, r \in i'j} \text{Var}[X_{ijq}] \text{Var}[X_{ijr}] \]

\[ = \frac{1}{\gamma^4} \sum_{k \in i'j} a_k^4 a_{kj}^4 [p_{ij}^4 (1 - p_{ij}) + (1 - p_{ij})^4 p_{ij}] \]

\[ + \frac{1}{\gamma^4} \sum_{q, r \in i'j} a_q^2 a_{jq}^2 p_{ij} (1 - p_{ij}) a_r^2 a_{jr}^2 p_{ij} (1 - p_{ij}) \]

\[ \leq \frac{1}{\gamma^4} \sum_{k \in i'j} a_k^4 a_{kj}^4 p_{ij} + \frac{1}{\gamma^4} \sum_{q, r \in i'j} a_q^2 a_{jq}^2 a_r^2 a_{jr}^2 p_{ij}^2 \]
and setting

\[ \gamma = 4C_1^2 \frac{n}{\gamma} \]

which gives us an absolute error bound between \( B \) and \( D^{-1}A^TA^{-1} \). It remains to get a relative error bound between \( DBD \) and \( A^TA \),

\[
\frac{\|DBD - A^T A\|}{\|A^T A\|} = \frac{\|D(B - D^{-1}A^T A^{-1}) D\|}{\|A^T A\|}
\]
by the submultiplicative property of the spectral norm,
\[ \leq \frac{\|D\| \|B - D^{-1} A^T A D^{-1}\|}{\|A^T A\|} \]

Now since $D$ is a diagonal matrix with positive entries, its spectral norm is its largest entry, i.e. the largest column magnitude, call it $c_*$,
\[ \leq c_*^2 \frac{\|B - D^{-1} A^T A D^{-1}\|}{\|A^T A\|} \]

Now we use another property of the spectral norm to lowerbound $\|A^T A\|$,
\[ \|A^T A\| = \max_{x,y \in \mathbb{R}^n} \frac{x^T A^T A y}{\|x\| \|y\|} \]

Setting $x, y$ to be indicator vectors to pick out the $i$’th diagonal entry of $A^T A$, we have that $\|A^T A\| \geq c_*^2$ since $c_*^2$ is some entry in the diagonal of $A^T A$. In addition to allowing us to bound the fourth central moment, this is yet another reason why we picked the sampling probabilities in Algorithm 3. Finally, continuing from above armed with this lower bound,
\[ \frac{\|DBD - A^T A\|}{\|A^T A\|} \leq \frac{c_*^2 \|B - D^{-1} A^T A D^{-1}\|}{\|A^T A\|} \leq \frac{c_*^2}{\|A^T A\|} \leq \frac{c_*^2}{c_*^2} = \epsilon \]

with probability at least $1/2$.

Although we had to set $\gamma = \Omega(n/\epsilon^2)$ to estimate the singular values, if instead of the singular values we are interested in individual entries of $A^T A$ that are large, we can get away setting $\gamma$ significantly smaller, and thus reducing shuffle size. In particular if two columns have high cosine similarity, we can estimate the corresponding entry in $A^T A$ with much less computation. Here we define cosine similarity as the normalized dot product
\[ \cos(c_i, c_j) = \frac{c_i^T c_j}{\|c_i\| \|c_j\|} \]

**Theorem 4.3.** For any two columns $c_i$ and $c_j$ having $\cos(c_i, c_j) \geq \epsilon$, let $B$ be the output of DIMSUMP with entries $b_{ij} = \frac{1}{\gamma} \sum_{k=1}^m X_{ijk}$ with $X_{ijk}$ as defined in Theorem 4.2. Now if $\gamma = \Omega(\alpha/\epsilon)$, 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
\[ \Pr \left[ \|c_i\| \|c_j\| \|b_{ij}\| < (1 - \delta) \|A^T A\|_{ij} \right] < \exp(-\alpha \delta^2 / 2) \]

**Proof.** We use $\|c_i\| \|c_j\| \|b_{ij}\|$ as the estimator for $\|A^T A\|_{ij}$. Note that
\[ \mu_{ij} = \mathbb{E} \left[ \sum_{k=1}^m X_{ijk} \right] = \gamma \frac{c_i^T c_j}{\|c_i\| \|c_j\|} = \gamma \cos(x, y) \geq \alpha \]

Thus by the multiplicative form of the Chernoff bound,
\[ \Pr \left[ \|c_i\| \|c_j\| \|b_{ij}\| > (1 + \delta) \|A^T A\|_{ij} \right] = \Pr \left[ \gamma \frac{\|c_i\| \|c_j\|}{\|c_i\| \|c_j\|} \|b_{ij}\| > \gamma (1 + \delta) \|A^T A\|_{ij} \right] \]
\[
\Pr \left[ \sum_{k=1}^{m} X_{ijk} > (1 + \delta) \mathbb{E} \left[ \sum_{k=1}^{m} X_{ijk} \right] \right] \leq \left( \frac{e^\delta}{(1 + \delta)^{1+\delta}} \right)^\alpha
\]

Similarly, by the other side of the multiplicative Chernoff bound, we have

\[
\Pr \left[ \|c_i\| \|c_j\| b_{ij} < (1 + \delta) [A^T A]_{ij} \right] = \Pr \left[ \left( \gamma \frac{\|c_i\| \|c_j\| b_{ij}}{\|\|c_i\| \|c_j\|} \right) \frac{\|c_i\| \|c_j\|}{\|b_{ij}\|} < \gamma (1 + \delta) [A^T A]_{ij} \right]
\]

\[
= \Pr \left[ \sum_{k=1}^{m} X_{ijk} < (1 + \delta) \mathbb{E} \left[ \sum_{k=1}^{m} X_{ijk} \right] \right] < \exp(-\mu_{ij} \delta^2 / 2) \leq \exp(-a \delta^2 / 2)
\]

5 Shuffle Size

Define \( H \) as the smallest nonzero entry of \( A \) in magnitude, after the entries of \( A \) have been scaled to be in \([0, 1]\). For example when \( A \) is a 0-1 matrix, \( H = 1 \).

**Theorem 5.1.** Let \( A \) be an \( m \times n \) tall and skinny \((m > n)\) sparse matrix \( A \) with at most \( L \) nonzeros per row. The expected shuffle size for DIMSUMMapper is \( O(nL\gamma/H^2) \).

**Proof.** Define \#(\( c_i, c_j \)) as the number of dimensions in which \( c_i \) and \( c_j \) are both nonzero, i.e. number of \( k \) for which \( a_{ki} a_{kj} \) is nonzero.

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

\[
\sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=1}^{n} \Pr[\text{DIMSUMMapper}(c_i, c_j)]
\]

\[
= \sum_{i=1}^{n} \sum_{j=i+1}^{n} #(c_i, c_j) \Pr[\text{DIMSUMMapper}(c_i, c_j)]
\]

By the Arithmetic-Mean Geometric-Mean inequality,

\[
\leq \frac{\gamma}{2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} #(c_i, c_j) \left( \frac{1}{\|c_i\|^2} + \frac{1}{\|c_j\|^2} \right)
\]

\[
\leq \gamma \sum_{i=1}^{n} \frac{1}{\|c_i\|^2} \sum_{j=1}^{n} #(c_i, c_j)
\]

\[
\leq \gamma \sum_{i=1}^{n} \frac{1}{\|c_i\|^2} L \|c_i\|^2 / H^2 = \gamma Ln / H^2
\]

The first inequality holds because of the Arithmetic-Mean Geometric-Mean inequality applied to \( \{1/\|c_i\|, 1/\|c_j\|\} \). The last inequality holds because \( c_i \) can co-occur with at most \( \|c_i\|^2 L / H^2 \) other columns. It is easy to see via Chernoff bounds that the above shuffle size is obtained with high probability.

**Theorem 5.2.** Let \( A \) be an \( m \times n \) tall and skinny \((m > n)\) sparse matrix \( A \) with at most \( L \) nonzeros per row. The shuffle size for any algorithm computing those entries of \( A^T A \) for which \( \cos(i,j) \geq \epsilon \) is at least \( \Omega(nL) \).
Proof. To see the lowerbound, we construct a dataset consisting of \( n/L \) distinct rows of length \( L \), furthermore each row is duplicated \( L \) times. To construct this dataset, consider grouping the columns into \( n/L \) groups, each group containing \( L \) columns. A row is associated with every group, consisting of all the columns in the group. This row is then repeated \( L \) times. In each group, it is trivial to check that all pairs of columns have cosine similarity exactly 1. There are \( \binom{L}{2} \) pairs for each group and there are \( n/L \) groups, making for a total of \( (n/L)\binom{L}{2} = \Omega(nL) \) pairs with similarity 1, and thus also at least \( \epsilon \). Since any algorithm that purports to accurately calculate highly-similar pairs must at least output them, and there are \( \Omega(nL) \) such pairs, we have the lower bound.

Finally it is easy to see that the largest reduce-key will have at most \( O(\gamma) \) values.

**Theorem 5.3.** The expected number of values mapped to a single key by DIMSUMMapper is \( \gamma \).

Proof. Note that the output of DIMSUMReducer is a number between 0 and 1. Since this is obtained by normalizing the sum of all values reduced to the key by \( \gamma \), and all summands are at most 1, we trivially get that the number of summands is at most \( \gamma \).

6 Conclusions and Future Directions

We presented the DIMSUM algorithm to compute \( A^T A \) for a very tall and skinny \( m \times n \) matrix \( A \). All of our results are provably independent of the dimension \( m \), meaning that apart from the initial cost of trivially reading in the data, all subsequent operations are independent of the dimension, the dimension can thus be very large.

Although we used \( A^T A \) in the context of computing singular values, there are likely other linear algebraic quantities that can benefit from having a provably efficient and accurate MapReduce implementation of \( A^T A \). For example if one wishes to use the estimate for \( A^T A \) in solving the normal equations in the ubiquitous least-squares problem

\[
A^T A x = A^T y
\]

then the guarantee given by Theorem 4.2 gives some handle on the problem, although a concrete error bound is left for future work.

7 Acknowledgements

We thank the Twitter Personalization and Recommender systems team for allowing us to use production data from the live Twitter site for experiments (not reported). We also thank Jason Lee, Yuekai Sun, and Ernest Ryu from Stanford ICME for valuable discussions. Finally we thank the Stanford student group: Computational Consulting and all its members for their help.
References


