Fast Monte-Carlo Algorithms for Approximate Matrix Multiplication

Petros Drineas*
Computer Science Department
Yale University
New Haven, CT 06520
drineas@cs.yale.edu

Ravi Kannan †
Computer Science Department
Yale University
New Haven, CT 06520
kannan@cs.yale.edu

Abstract

Given an $m \times n$ matrix A and an $n \times p$ matrix B, we present 2 simple and intuitive algorithms to compute an approximation P to the product $A \cdot B$, with provable bounds for the norm of the "error matrix" $P - A \cdot B$. Both algorithms run in $O(mp+mn+np)$ time. In both algorithms, we randomly pick $s = O(1)$ columns of A to form an $m \times s$ matrix S and the corresponding rows of B to form an $s \times p$ matrix R. After scaling the columns of S and the rows of R, we multiply them together to obtain our approximation P. The choice of the probability distribution we use for picking the columns of A and the scaling are the crucial features which enable us to give fairly elementary proofs of the error bounds. Our first algorithm can be implemented without storing the matrices A and B in Random Access Memory, provided we can make two passes through the matrices (stored in external memory). The second algorithm has a smaller bound on the 2-norm of the error matrix, but requires storage of A and B in RAM. We also present a fast algorithm that "describes" P as a sum of rank one matrices if $B = A^T$.

1 Introduction

In many applications the data consists of a large $m \times n$ matrix A and it is of interest to compute (or approximate) the product $A \cdot A^T$.

One such application is to Information Retrieval. Given a database of documents, we represent it as an $m \times n$ document-term matrix A (that is m documents each described w.r.t. n terms or equivalently m document-vectors in $\mathcal{R}^n$). Our goal is to find all document - document matches, that is all entries in $A \cdot A^T$ that are larger than a certain threshold (see for example [4]). Obviously, straight-forward algorithms for computing the product $A \cdot A^T$ might not be efficient in this case, especially since exact values for the elements of $A \cdot A^T$ are not necessary.

We use the following notation : for any matrix X, we denote by $X^{(i)}$ the i th row of X as a row vector and by $X_{(i)}$ the i th column of X as a column vector. Then, note that we may write AB as the sum of n rank one matrices :

$$AB = \sum_{t=1}^{n} A^{(t)} B_{(t)}. \tag{1}$$

¿From this, a simple algorithm for approximate matrix multiplication suggests itself : pick a random subset of s columns of A to form an $m \times s$ matrix S; form an $s \times p$ matrix R out of the **corresponding** columns of B. Then, intuitively, it follows from (1) that the product SR is an estimator (entry by entry) of the product AB; the variance remains to be worked out. Our contribution to the above (following the lines of [5] and [7]) is two-fold. First, instead of picking columns uniformly at random, we pick them according to some "more interesting" probability distribution. In general, we pick a column with probability proportional to its length squared, which is a measure of the amount of "information" the column contains. Second, before including a column in the sample, we scale it in order to compensate for the columns that are not picked. With these two improvements, interesting bounds can be proven for the error of the approximations.

This approach for approximating matrix multiplication has obvious advantages. It is conceptually simple, it can be easily implemented and it can be generalized to approximate the product of more than 2 matrices. Also, since the "heart" of the algorithm involves matrix multiplication of smaller matrices, it can use any algorithms that exist in the literature for performing the desired matrix multiplication. Another advantage

*Supported by NSF Grant CCR-9820850.
†Supported in part by NSF Grant CCR-9820850.

is that it does not tamper with the sparsity of the matrices, unlike the following folklore algorithm on lines similar to ours : project both A and B to the same random s dimensional subspace and take the product of the projections. Finally, we will see that in one pass through the matrices (one sequential read, say, from external memory), we can draw the required sample of the columns of A and then, in another pass, we can construct S and R. So, we only need Random Access Memory to store S and R, not A and B.

In the following theorems we are proving bounds for $\|P-AB\|_F$ and $|P-AB|_2$ (Notation : $\|A\|_F^2 = \sum_{i,j} A_{ij}^2$ and $|A|_2 = \max|Ax|, \forall x \in \mathcal{R}^n$ s.t. $|x| = 1$). These are the two traditional norms we use - the F stands for Frobenius norm and $|A|_2$ is called the 2-norm).

Theorem 1 *For any fixed $s > 0$ there is a randomized algorithm that approximates the product $A \cdot B$ by a matrix P in $O(smp)$ time such that,*

$$E(\|P - A \cdot B\|_F^2) \leq \frac{1}{s}\left(\sum_{k=1}^{n} |A_{(k)}||B^{(k)}|\right)^2 \leq \frac{1}{s}||A||_F^2||B||_F^2$$

Further, the algorithm makes two passes through the matrices A and B and only uses RAM space $O(s(m+p))$ provided it has a write-only output tape to write P out.

After reading both matrices once, the algorithm needs $s(m+p)$ space to store the columns and rows that are kept. In section 3 we show that for certain matrices we can avoid even reading the matrices once with some loss in accuracy. In section 4 we present element-wise error bounds for the approximation.

For matrices such that $\sum_{k=1}^n |A_{(k)}||B^{(k)}|$ is close to $\|AB\|_F$, the relative error of the approximation is inversely proportional to the square root of the number of columns that are picked. For example, this will happen if $||AB||_F$ is $\Omega(||A||_F||B||_F)$. It is well-known that $||AB||_F \leq ||A||_F||B||_F$. So, $||AB||_F$ is $\Omega(||A||_F||B||_F)$ if there is "not much cancellation" when we do the multiplication of A and B. Also, in the special case where $B = A^T$, it is easy to see that if A is a matrix of small rank or is well-approximated by a matrix of small rank, then $||AA^T||_F \in \Omega(||A||_F^2)$ and in this case, indeed, we get a good approximation to AA^T. Unfortunately, in general we cannot guarantee that the above quantities are close and thus we can not get relative error bounds. Theorem 1 also implies a bound for $|P-AB|_2$, since we know that the 2-norm of any matrix is bounded above by the Frobenius norm.

A different algorithm with the same running time but requiring $O(mn+np)$ storage provides a better 2-norm bound. The following theorem is proved in section 5 using an extension of the Furedi-Komlos result for the top eigenvalue of random symmetric matrices to the unsymmetric case (see [8], [1]).

Theorem 2 *For any constant s there exists a randomized algorithm that approximates the product $A \cdot B$ by some matrix P in $O(smp)$ time such that, with probability at least $1-(m+n)^{-1}$*

$$|P - AB|_2 \leq (7/3) \cdot nM^2\sqrt{m+p}/\sqrt{s}$$

assuming $s \leq \frac{m+p}{11^6 \ln^6(m+p)}$[1] and $|A_{ij}| \leq M, |B_{ij}| \leq M$ for all i, j and some positive $M \in \mathcal{R}$.

To gain some intuition on the above error bound, assume $m = n = p$. Observe that there exists $\tau \leq 1$ such that $\|AB\|_F = \tau n^2 M^2$. Also, $|AB|_2 \geq \|AB\|_F/\sqrt{n} \geq \tau n^{3/2} M^2$. Thus, the relative error of our approximation w.r.t. the 2-norm is at most $\frac{3.3\tau}{\sqrt{s}}$. This analysis of course is useful only if τ is close to 1, e.g. if there is a lot of correlation between the rows of A and the columns of B. However, note that using the bounds implied by Theorem 1, even if τ is close to 1, we don't get a relative error bound w.r.t. the 2-norm.

To the best of our knowledge, the only previous randomized algorithm that approximates the product of two matrices appears in [4]. Their algorithm is based on random walks in a graph representation of the input matrices. It is more complicated, requires different graph representations of the input matrices if they are allowed to contain negative entries and it needs to fully store the input matrices. A preliminary theoretical comparison between the two algorithms appears in section 4.

If we are seeking to approximate AA^T, instead of explicitly computing all elements of P (a task that takes $O(m^2)$ time already) we can "describe" P faster, by providing column vectors $\bar{p}^{(t)} \in \mathcal{R}^m, t = 1 \ldots s$ such that $P = \sum_{t=1}^{s} \bar{p}^{(t)} \bar{p}^{(t)^T}$. The $\bar{p}^{(t)}$ form an orthogonal (but not orthonormal) set of vectors. This theorem is based on older results ([5],[7]) that approximate the Singular Value Decomposition of a matrix. Reconstructing P from its "description" would take $O(m^2)$ time (see section 6).

2 The matrix multiplication algorithm

Suppose A is an $m \times n$ matrix, B an $n \times p$ matrix and we want to approximate the product $A \cdot B$. An

[1] In the Appendix we show how to improve this constant using stronger concentration results.

obvious algorithm would be to pick at random a subset of s columns of A to form an $m \times s$ matrix S and the corresponding s rows of B to form an $s \times p$ matrix R. Then, we return $S \cdot R$ as an approximation to $A \cdot B$. This approach has been proposed, for example, in [4], but, as the authors comment, the variance of the approximation (elementwise) is too high for the algorithm to be useful.

We modify this algorithm to incorporate the non uniform sampling and scaling that we proposed in the introduction. Suppose we have :

$$p_1, p_2, \ldots p_n \geq 0 \text{ such that } \sum_{k=1}^{n} p_k = 1.$$

- **for $t = 1$ to s independently**
  - Pick $i_t \in \{1 \ldots n\}$ at random with
    $$\text{Prob}(i_t = k) = p_k, \quad k = 1 \ldots n.$$
  - Include $A_{(i_t)}/\sqrt{sp_{i_t}}$ as a column of S and $B^{(i_t)}/\sqrt{sp_{i_t}}$ as the corresponding row of R.
- Return $S \cdot R$ as the approximation to $A \cdot B$.

Our first lemma proves that the expectation of the ij-th element of the approximation is equal to the ij-th element of the exact product. The second lemma describes the variance of the approximation.

Lemma 1 *Given the above definitions,* $E((SR)_{ij}) = (AB)_{ij}$.

Proof: Fix attention on one particular i, j. For $t = 1 \ldots s$ define the random variable $X_t = \left(\frac{A_{(i_t)}B^{(i_t)}}{sp_{i_t}}\right)_{ij} = \frac{A_{ii_t}B_{i_tj}}{sp_{i_t}}$. So, the X_t's are independent random variables. Also, $(SR)_{ij} = \sum_{t=1}^{s} X_t$. Thus, its expectation is equal to the sum of the expectations of the X_t's. But, $E(X_t) = \sum_{k=1}^{n} \frac{A_{ik}B_{kj}}{sp_k} p_k = \frac{1}{s}(AB)_{ij}$. So, $E((SR)_{ij}) = \sum_{t=1}^{s} E(X_t) = (AB)_{ij}$. $\diamond$

Lemma 2 *Given the above definitions,*

$$Var((SR)_{ij}) = \frac{1}{s} \sum_{k=1}^{n} \frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{s}(AB)_{ij}^2$$

Proof: Since $(SR)_{ij}$ is the sum of s independent random variables, the variance of $(SR)_{ij}$ is the sum of the variances of these variables. But, using $\text{Var}(X_t) = E(X_t^2) - E^2(X_t)$ we see that $\text{Var}(X_t) = \sum_{k=1}^{n} \frac{A_{ik}^2 B_{kj}^2}{s^2 p_k} - \frac{1}{s^2}(AB)_{ij}^2$ and the lemma follows. $\diamond$

Lemma 3 *If* $p_k = \frac{|A_{(k)}||B^{(k)}|}{\sum_{k=1}^{n} |A_{(k)}||B^{(k)}|}$, *then* $\sum_{k=1}^{n} p_k = 1$ *and*

$$E(||AB - SR||_F^2) = \frac{1}{s}\left(\sum_{k=1}^{n} |A_{(k)}||B^{(k)}|\right)^2 - \frac{1}{s}\|AB\|_F^2.$$

This choice for p_k minimizes the variance of the error of the approximation.

Proof: For part 1, using lemmas 1 and 2,

$$E(\|AB - SR\|_F^2) = \sum_{i=1}^{m}\sum_{j=1}^{n} Var((SR)_{ij}) =$$
$$\frac{1}{s}\sum_{k=1}^{n} \frac{1}{p_k}(\sum_i A_{ik}^2)(\sum_j B_{kj}^2) - \frac{1}{s}\|AB\|_F^2 =$$
$$\frac{1}{s}\sum_{k=1}^{n} \frac{1}{p_k}|A_{(k)}|^2|B^{(k)}|^2 - \frac{1}{s}\|AB\|_F^2 =$$
$$\frac{1}{s}\left(\sum_{k=1}^{n} |A_{(k)}||B^{(k)}|\right)^2 - \frac{1}{s}\|AB\|_F^2$$

To prove that this is the minimal upper bound for the variance that we can get with any choice of the p_k's we define the function (observe that $\frac{1}{s}\|AB\|_F^2$ is independent of the p_k)

$$f(p_1, \ldots p_n) = \sum_{k=1}^{n} \frac{1}{p_k}|A_{(k)}|^2|B^{(k)}|^2$$

We want to minimize f given that $\sum_{k=1}^{n} p_k = 1$. Using simple calculus (that is substituting $p_n = 1 - \sum_{k=1}^{n-1} p_k$ and solving the system of equations $\frac{\partial f}{\partial p_i} = 0, i = 1, \ldots n-1$ we get the above p_k's. $\diamond$

If we are interested in the multiplication $A \cdot A^T$ the above result becomes

Lemma 4 *If* $p_k = \frac{|A_{(k)}|^2}{\|A\|_F^2}$, *then* $\sum_{k=1}^{n} p_k = 1$ *and*

$$E(\|AA^T - SS^T\|_F^2) \leq \frac{1}{s}\|A\|_F^4 - \frac{1}{s}\|AA^T\|_F^2$$

This choice for p_k minimizes the variance of the error of the approximation.

Now we analyze the running time and space requirements of the algorithm. The algorithm works on matrices in sparse representation, where the matrix is presented as a set of triples (i, j, A_{ij}) with at most one triple for each (i, j). So, the non-zero entries need not be given; some 0 entries may be presented.

Lemma 5 *Suppose A and B are presented in sparse representation. Then in one pass through A and B from external memory, we can compute $|A_{(k)}|, |B^{(k)}|, k = 1, 2, \ldots n$ and p_k as defined in lemma 3. In a second pass, we can form the matrices S and R defined in the algorithm in RAM and compute SR. If the maximum number of entries in any column of A (in the sparse representation) is q and the maximum number of entries in any row of B is r, then the total RAM required is $O(s(q + r))$. The total time required is the time for the two passes (and the computation of the p_k) plus $O(s(q + r) + s \cdot \min(mq, pr))$.*

Proof: It is obvious that the p_k can be computed on the first pass. It is also clear that the entries required for S and R can be pulled out in the second pass. Then we do the required scaling. To perform the multiplication of S and R (assuming $pr \leq mq$), we note that for multiplying each row of S by R, we need to make just one pass through all the (presented) entries of R. This can be done in time $O(pr)$. If $mq \leq pr$, then for multiplying S by each column of R we would make one pass through S. $\diamond$

If the matrices are not in sparse representation, then we may use the fast matrix multiplication algorithms for multiplying dense matrices S and R (see [14],[3]).

3 Sampling with near-optimal probabilities

There might be applications where making two passes through the matrix is not possible. For example, this is the case when the input matrices are large data streams that we cannot store in memory (in the "streaming model" of [6] only one pass is allowed). The question arises : can we do something in just one pass through the matrices? One important thing we can do in one pass is uniform sampling - i.e., we can choose uniformly at random s integers from $\{1, 2, \ldots n\}$ (before reading any data.) Then, in one pass, we may form the matrix S consisting of the chosen columns of A and the matrix R consisting of the corresponding rows of B and multiply them. Under some conditions, this yields good error bounds, as will follow from lemma 6 below.

This lemma says that we can essentially get a result like Theorem 1 even if the probabilities we use are not the optimal ones given in lemma 3. Besides giving us some information in the case of uniform sampling (as discussed below), it also helps in the situation when we have apriori knowledge of the matrices A and B and know estimates of $|A_{(k)}|$ and $|B^{(k)}|$.

Lemma 6 *For any set of probabilities $q_k, k = 1 \ldots n$ such that $q_k \geq c \cdot p_k = \frac{c|A_{(k)}||B^{(k)}|}{\sum_{k=1}^{n} |A_{(k)}||B^{(k)}|}$ (for some positive constant $c \leq 1$),*

$$E(\|A \cdot B - S \cdot R\|_F^2) \leq \frac{1}{cs}\left(\sum_{k=1}^{n} |A_{(k)}||B^{(k)}|\right)^2$$

Proof: Similar to the proof of lemma 3. $\diamond$

So, assume that for matrices A and B, we can guarantee that for all $k = 1 \ldots n, |A_{(k)}||B^{(k)}|$ is close to its mean value $(\frac{1}{n}\sum_{k=1}^{n} |A_{(k)}||B^{(k)}|)$. More precisely, assume that there exists some positive constant $c \leq 1$ such that for all $k = 1 \ldots n$, $|A_{(k)}||B^{(k)}| \leq \frac{c^{-1}}{n}\sum_{k=1}^{n} |A_{(k)}||B^{(k)}|$. This implies that $1/n \geq c \cdot p_k$ for every k. Thus, we could perform uniform sampling with a small loss in accuracy (depending on c).

So far we analyzed our algorithm assuming that the sampling of the columns (and rows) is done *with replacement*. One interesting variant would be to examine the error bound of our algorithm if sampling *without replacement* is performed.

Lemma 7 *If $p_k = 1/n$ and sampling is performed without replacement, then, with probability at least $1 - \delta$,*

$$\|A \cdot B - S \cdot R\|_F \leq \sqrt{\frac{n}{(n-1)\delta}\left(\frac{n}{s} - 1\right)\sum_{t=1}^{n} |A_{(k)}|^2 |B^{(k)}|^2}$$

$$\approx \sqrt{\frac{1}{\delta}\left(\frac{n}{s} - 1\right)\sum_{t=1}^{n} |A_{(k)}|^2 |B^{(k)}|^2}$$

Scetch of proof: We compute the expectation and the variance of $(SR)_{ij}$ from first principles. The proof is straightforward but quite tedious. $\diamond$

We see that for $s = n$ the above error becomes zero, while in Lemma 4 this was not the case. Unfortunately, for the weighted sampling case, a similar analysis is rather hard, because the scaling factors that the algorithm uses become very complicated. If we keep the scaling factors simple the expectation of $(SR)_{ij}$ is not equal to $(AB)_{ij}$ and the error of the approximation blows up.

4 Element-wise error bounds

In this section we provide element-wise error bounds for our algorithm.

Theorem 3 *Assuming uniform sampling ($p_k = \frac{1}{n}$) and $|A_{ij}| \leq M, |B_{ij}| \leq M$, for any $\delta > 0$, the following holds with probability at least $1-\delta$*

$$|(AB)_{ij} - (SR)_{ij}| \leq \frac{nM^2}{\sqrt{s}}\sqrt{2\ln(mp) + 2\ln(2/\delta)}, \forall i, j$$

Proof: Fix attention on one particular i, j. Define $X_t = \left(\frac{A_{(i_t)}B^{(i_t)}}{sp_{i_t}}\right)_{ij} = \frac{A_{ii_t}B_{i_tj}}{sp_{i_t}}$. Then, as in lemma (1, we have that the expectation of X_t is $\frac{1}{s}(AB)_{ij}$ and so defining $Y_t = \frac{1}{s}(AB)_{ij} - X_t, t = 1 \ldots s$, we have that the Y_t's are independent random variables and $E(Y_t) = 0$. Also,

$$|Y_t| = \left|\frac{1}{s}(AB)_{ij} - \frac{A_{ii_t}B_{i_tj}}{sp_{i_t}}\right| \leq \frac{2n}{s}M^2$$

Now we use Theorem 2 from [10] which bounds the probability of the sum of bounded random variables deviating from the mean. For any $t > 0$,

$$\text{Prob}\left(\left|\sum_{t=1}^{s} Y_t\right| \geq st\right) \leq 2e^{-\frac{2s^2t^2}{s\cdot 4n^2M^4s^{-2}}} = 2e^{-\frac{2s^3t^2}{4n^2M^4}}.$$

Now, setting $t = \frac{nM^2\sqrt{2\ln(mp)+2\ln(2/\delta)}}{s^{3/2}}$ we get an upper bound of $\delta/(mp)$ on the probability of failure for one particular i, j. We get the theorem by multiplying this by mp, the number of (i, j). $\diamond$

It is obvious from the above error bound that for entries of AB that are "large" (that is close to the maximum value nM^2), we get approximations with small relative error. So, the algorithm looks appropriate for our purposes: it returns more accurate approximations to the largest entries!

The above analysis yields essentially the same results as the uniform sampling algorithm of section 3, but it improves the probability with which they hold. Ignoring logarithmic factors (which are involved because of the higher probability with which this result holds), Theorem 3 guarantees that *every* entry of the approximation will have additive error at most $nM^2/\sqrt{s}$ with high probability. Thus, with the same probability, $\|AB - SR\|_F^2 \leq mpn^2M^4/s$. The bound of lemma 6 is generally tighter, although it is upper bounded by the same quantity. The reason is that we now expect *every* element of the product matrix to be approximated within a fixed additive error. Lemma 6 provides a tighter Frobenius norm bound, but not element-wise guarantees.

For clarity and simplicity, we presented bounds only for the uniform sampling case. We could use weighted sampling and avoid using the crude upper bound M, to make the above results tighter.

We can compare the above error bounds to the results of [4]. We will do the comparison only for matrices with positive elements [2]. There, the number of samples [3] needed to guarantee approximations to every entry of AB within an additive error equal to the one of Theorem 3 is proportional to $\frac{\sum_{i=1}^{m}(M_i)^2}{s^{-1}n^2M^4}$, where M_i is the sum of the elements across the i-th row of the product AB. Estimating this ratio is not easy, but we can see that even if one of the M_i's is sufficiently close to pnM^2 (the maximum value of any M_i) the running time becomes $O(sp^2)$. This is comparable to the running time of our algorithm. If more of the M_i's are large our algorithm performs better.

5 A second algorithm with a better 2-norm bound

We will now analyze an algorithm that estimates each entry of the product $A \cdot B$ independently. So, in order to estimate $(AB)_{ij}$ we pick s elements from the i-th row of A and the corresponding s elements from the j-th row of B, scale them and return the sum of their products as an approximation to $(AB)_{ij}$.

The algorithm is:

- **for all** $i = 1 \ldots m, j = 1 \ldots p$ **independently**
    1. **for** $t = 1$ **to** s **independently**
        - Pick an integer $i_t \in \{1 \ldots n\}$ (with replacement), where $\text{Prob}(i_t = k) = p_k, k = 1 \ldots n$.
        - Compute $X_t^{ij} = \frac{A_{ii_t}B_{i_tj}}{sp_{i_t}}$.
    2. Return $\sum_{t=1}^{s} X_t^{ij}$ as the approximation to $(AB)_{ij}$.

Following the ideas of section 2, we may prove bounds for the Frobenius norm of the error matrix. Since $E(P_{ij}) = (AB)_{ij}$ and $\text{Var}(P_{ij}) \leq \frac{1}{s}\sum_{k=1}^{n}\frac{A_{ik}^2B_{kj}^2}{p_k}$, we see that

1. If $p_k = \frac{1}{n}$, $E(\|P - AB\|_F^2) \leq \frac{n}{s}\sum_{k=1}^{n}|A_{(k)}|^2|B^{(k)}|^2$.
2. If $p_k = \frac{A_{ik}^2}{|A^{(i)}|^2}$, $E(\|P - AB\|_F^2) \leq \frac{1}{s}\|A\|_F^2\|B\|_F^2$. This is not the "optimal" sampling probability distribution (it does not minimize the variance of the error) but computing the optimal one would take $O(mnp)$ time.

[2] Similar results hold for matrices with negative elements as well.

[3] The algorithm performs $O(1)$ operations per sample, thus its running time is O(samples).

The second error bound is generally better. We could also obtain Chernoff bounds for the element-wise error of our approximations. We note here that if we perform uniform sampling, the running time of the algorithm is $O(smp)$. If non-uniform sampling is performed the running time becomes $O(smp\log n)$. The reason is that in order to approximate $(AB)_{ij}$ we need to sample s elements from the i th row of A w.r.t. their weights, which takes $s \log n$ time. Also, in both cases we need to fully store matrices A and B ($O(mn+np)$ space).

This algorithm returns much nicer error guarantees with respect to the 2-norm. In proving the following theorem we use a straight-forward extension of the classical Furedi-Komlos result for the top eigenvalue of a random symmetric matrix to general matrices (see [1]).

Theorem 4 *If we compute P using the above algorithm and uniform sampling, with probability at least $1-(m+p)^{-1}$,*

$$|P-AB|_2 \leq (7/3)nM^2\sqrt{m+p}/\sqrt{s}$$

assuming $s \leq \frac{m+p}{11^6 \ln^6(m+p)}$.

Proof: We observe that for all i,j, $(P-AB)_{ij}$ are zero mean independent random variables and $\mathrm{Var}((P-AB)_{ij}) = \sum_{k=1}^{n} \frac{A_{ik}^2 B_{kj}^2}{sp_k}$. Assuming uniform sampling ($p_k = 1/n$), $\mathrm{Var}((P-AB)_{ij}) \leq n^2M^4/s$.

So, using Theorem 4 of [1], for $a = 3/2$, with probability at least $1-(m+p)^{-1}$, $|P-AB|_2 \leq (7/3)\cdot \frac{nM^2}{\sqrt{s}}\sqrt{m+p}$. $\diamond$

The constraint on s is necessary in order to satisfy the condition of Theorem 4 of [1] and a similar condition in [8]. It can be improved (11^6 can be eliminated) by using better concentration results (Talagrand's inequality, see Appendix) and slightly increasing the constant $7/3$ in the error bound. Nevertheless, this restriction on s renders this result (as well as the result of [1]) useless for small matrices, since it restricts the maximum value of s (obviously, s must be at least one and in general larger in order to reduce the error). All our experimental evidence though suggests that the algorithm is useful in practice, even for small matrices, and that the constraint needs not be satisfied. Removing it is an open problem.

6 A different approach

Any $m \times n$ matrix A can be expressed as

$$A = \sum_{t=1}^{r} \sigma_t u^{(t)} {v^{(t)}}^T$$

where r is the rank of A, σ_t are its singular values and $u^{(t)}$ and $v^{(t)}$ are its left and right singular vectors respectively. We remind the reader that the $u^{(t)}$'s and the $v^{(t)}$'s form orthonormal sets of vectors. Then, if $p^{(t)} = \sigma_t u^{(t)}$,

$$A \cdot A^T = \sum_{t=1}^{r} p^{(t)} {p^{(t)}}^T$$

Exact computation of the $p^{(t)}$'s takes time $O(m^2n + mn^2)$. But, in [5] and [7], algorithms have been presented that compute "good" approximations $\bar{p}^{(t)}$ to the top s $p^{(t)}$'s fast. So, we can approximate $A \cdot A^T$ by P, where

$$P = \sum_{t=1}^{s} \bar{p}^{(t)} {\bar{p}^{(t)}}^T$$

6.1 The algorithm

In [5] we presented an algorithm that computes the $\bar{p}^{(t)}$'s and we proved that using them we get an accurate low-rank approximation to A. Here we adapt the same algorithm to approximate $A \cdot A^T$.

- Create the $m \times s$ matrix S as described in section 2.
- Compute $S^T \cdot S$ and its singular value decomposition.

  So, $S^T S = \sum_{t=1}^{s} \lambda_t^2 w^{(t)} {w^{(t)}}^T$, where λ_t are the singular values of S and $w^{(t)}, t = 1 \ldots s$ its right singular vectors. Thus, we can approximate $p^{(t)}, t = 1 \ldots s$ by the left singular vectors of S scaled by the corresponding singular values, namely $\bar{p}^{(t)} = Sw^{(t)}, t = 1 \ldots s$.
- "Describe" P (our approximation to $A \cdot A^T$) as $P = \sum_{t=1}^{s} \bar{p}^{(t)} {\bar{p}^{(t)}}^T$

The $\bar{p}^{(t)}$'s are orthogonal. Also, from basic properties of singular value decomposition, we observe that

$$\sum_{t=1}^{s} Sw^{(t)} {w^{(t)}}^T S^T = S \cdot S^T$$

Lemma 8 *If P is constructed as described in the above algorithm, with probability at least $1-\delta$,*

$$E\left(\|A \cdot A^T - P\|_F^2\right) = E\left(\|A \cdot A^T - S \cdot S^T\|_F^2\right) \leq \frac{1}{s}\|A\|_F^4$$

Lemma 9 *If we allow preprocessing time to read all non-zero elements of matrix A once, we can compute a "description" to P using the above algorithm in $O(s^2m + s^3)$ time.*

Proof: The computation of S^TS takes $O(s^2m)$ time and the computation of its singular value decomposition takes $O(s^3)$ time. The $\bar{p}^{(t)}, t = 1 \dots s$ can be computed in $O(s^2m)$ time. $\diamond$

6.2 Generalizing the SVD-based algorithm

To multiply an $m \times n$ matrix A by an $n \times p$ matrix B using the above algorithm, we define $C = \begin{bmatrix} A & \mathbf{0} \\ B^T & \mathbf{0} \end{bmatrix}$. Then $C \cdot C^T = \begin{bmatrix} A \cdot A^T & A \cdot B \\ B^T \cdot A^T & B^T \cdot B \end{bmatrix}$. The error now depends on $\|A\|_F^2 + \|B\|_F^2$ instead of $\sum_{k=1}^n |A_{(k)}||B^{(k)}|$ (see lemma 3). The later quantity is generally larger.

7 Experiments

We tested our algorithms using an information retrieval data set, namely a collection of 5000 encyclopedia articles. A popular technique in information retrieval is Latent Semantic Indexing (LSI), where A is replaced by an $m \times k$ matrix $\bar{A}$ (by projecting all m document-vectors to a fixed-dimensional space $\mathcal{R}^k$). This is to help reduce the "noise" (see [2] for a survey). This process can be implemented by computing the SVD of A. As in [4], we also dealt here not with the actual document-term matrix, but its projection to 320 dimensions.

We assumed that document-document matches are dot-products that are larger than some threshold τ and we tried to identify all document-document matches in the database, which is equivalent to computing AA^T and finding all entries that are larger than τ. In preliminary experiments using the algorithms of sections 2 and 3, we were able to achieve a speedup of 4 versus full matrix multiplication and at the same time identify more than 99% of the document-document matches for various values of τ. As an example, if $\tau = 0.85$, less than 0.04% of the $25 \cdot 10^6$ document-document pairs are considered to be matches, but we were still able to identify 99.4% of the matches.

8 Tightness of the results and open problems

We believe that the bound of the algorithm of section 2 with respect to the F-norm is tight (for this particular algorithm). Consider any $m \times n$ matrix $A = UV^T$, where U and V are square orthonormal matrices of appropriate dimensions. Then, the algorithm of section 2 returns an error of

$$\|AA^T - SS^T\|_F^2 \approx n^2/s$$

Scetch of proof: Obviously, $AA^T = UU^T$. Also, S is equal to $A \cdot T$, where T is an $n \times s$ matrix such that $T_{ij} = \sqrt{\frac{\|A\|_F^2}{s|A_{(i)}|^2}}$ if column i was included in the sample. But, for this particular A we can argue that the above ratio is almost $\sqrt{n/s}$ for all columns, since they all have approximately the same length. Thus, using basic Linear Algebra,

$$\|AA^T - SS^T\|_F^2 = \|UU^T - UV^TTT^TVU^T\|_F^2 =$$
$$\|I_n - TT^T\|_F^2 = (n-s) + s \cdot \left(\frac{n}{s} - 1\right)^2$$

For $s << n$ we see that $\|AA^T - SS^T\|_F^2 \approx n^2/s$. But, $\|A\|_F^2 = n$, thus Lemma 4 returns a tight bound. $\diamond$

We do not have a similar result for the 2-norm of the error. We suspect that the 2-norm bound could be slightly improved: namely,

$$E(\|AA^T - SS^T\|_2^2) \leq \|A\|_F^4/s^2$$

This improvement is still open.

The most interesting open problem would be to devise new randomized algorithms, with similar running time and space requirements, that provide relative error bounds or prove the lack thereof. So far, the algorithm in section 5 has the best error bounds. Unfortunately, even incorporating weighted sampling in this algorithm and strengthening the result of [8] to account for different element-wise variances would not return a relative error guarantee (in general), but it would return a tighter bound by avoiding the use of the crude upper bound M. Finally, it would be interesting to achieve the same error bound with respect to the 2-norm, while using only linear space.

Acknowledgements: We would like to thank Dimitris Achlioptas for bringing to our attention the results of [12] and [13] and David Lewis for providing us the corpus for the experiments of section 7.

References

[1] D. Achlioptas and F. McSherry, *Fast Computation of Low Rank Approximations*, Proceedings of the 33rd Annual Symposium on Theory of Computing, 2001.

[2] M.W.Berry, S.T.Dumais and G.W.O'Brien, *Using linear algebra for intelligent information retrieval*, SIAM Review, 37(4), 1995, pp. 573-595.

[3] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, J. Symbolic Comput. 9 (1990), no. 3, pp. 251-280.

[4] E. Cohen and D. Lewis, *Approximating matrix multiplication for pattern recognition tasks*, J. Algorithms 30 (1999), no. 2, pp. 211-252. Preliminary version in the Proceedings of the 8th Symposium on Discrete Algorithms, 1997.

[5] P. Drineas, A. Frieze, R. Kannan, S. Vempala and V. Vinay, *Clustering in large graphs and matrices*, Proceedings of the 10th Symposium on Discrete Algorithms, pp. 291-299, 1999.

[6] J. Feigenbaum, S. Kannan, M. Strauss and M. Vishwanathan, *An Approximate l^1-differnce algorithm for massive data sets*, Proceedings of the 40th Annual IEEE Symposium on the Foundations of Computer Science, pp. 501-511, 1999.

[7] A. Frieze, R. Kannan and S. Vempala, *Fast Monte-Carlo algorithms for finding low rank approximations*, Proceedings of the 39th Annual Symposium on Foundations of Computing, pp. 370-378, 1998.

[8] Z. Furedi and J. Komlos, *The eigenvalues of random symmetric matrices*, Combinatorica 1 (1981), pp. 233-241.

[9] G.H.Golub and C.F.Van Loan, *Matrix Computations*, Johns Hopkins University Press, London, 1989.

[10] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, American Statistical Association Journal, March 1962, pp. 13-30.

[11] F. Jiang, R. Kannan, M. Littman and S. Vempala, *Efficient singular value decomposition via improved document sampling*, manuscript, 1999.

[12] M. Krivelevich and V. Vu, *Approximating the independence number and the chromatic number in expected polynomial time*, Automata, languages and programming, pp. 13-24, Geneva 2000.

[13] M. Krivelevich and V. Vu, *On the concentration of eigenvalues of random symmetric matrices*, Technical Report MSR-TR-2000-60, 2000.

[14] V. Strassen, *Gaussian elimination is not optimal*, Numer. Math. 13, pp. 354-356, 1969.

[15] J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, New York, 1965.

Appendix

We will describe a way of improving the restriction on s, with a slight loss in accuracy (see section 5). We follow the lines of [1], [12] and [13]. Start by defining the matrix $F = (2nM^2)^{-1} \cdot \begin{bmatrix} \mathbf{0} & (P-AB)^T \\ P-AB & \mathbf{0} \end{bmatrix}$, which is a $(m+p) \times (m+p)$ symmetric matrix, $|F_{ij}| \leq 1$ and $E(F_{ij}) = 0$ for all i, j. We also note that $\sigma^2 = \text{Var}(F_{ij}) = (2nM^2)^{-2}\text{Var}((P-AB)_{ij}) \leq (4s)^{-1}$.

We denote by λ_i, $i = 1 \ldots r$ the eigenvalues of F (r is the rank of F) and let λ denote the $\max_i |\lambda_i|$. Assuming k is a positive even integer, $\text{Tr}(F^k) = \sum_{i=1}^r \lambda_i^k$, thus $E(\lambda^k) \leq E(\text{Tr}(F^k))$. In [8], it is proven that $E(\text{Tr}(F^k)) \leq (m+p)^{k/2+1} 2^k \sigma^k$. Thus, $E(\lambda) \leq 2\sigma(m+p)^{1/k}\sqrt{m+p} \leq \sqrt{\frac{m+p}{s}}(m+p)^{1/k}$. There is a restriction on k, namely $k^3 \leq \sigma\sqrt{m+p}$ or, equivalently, $s \leq \frac{m+p}{4k^6}$.

We will now use a theorem proven in [13] through a powerful concentration result (Talagrand's inequality). Let q denote the *median* of λ. For any $t > 0$,

$$\text{Prob}(|\lambda - q| \geq t) \leq 4e^{-t^2/32}$$

But, in [13] it is also proven that

$$|E(\lambda) - q| \leq 64$$

Combining the above results, it is straight-forward to see that, with probability at least $1 - 4e^{-t^2/32}$,

$$|\lambda - E(\lambda)| \leq t + 64$$

Setting $t = \sqrt{32 \ln 4(m+p)}$, we get that with probability at least $1 - (m+p)^{-1}$

$$\lambda \leq (m+p)^{1/k}\sqrt{\frac{m+p}{s}} + \sqrt{32\ln 4(m+p)} + 64$$

Obviously, $|P - AB|_2 = 2nM^2\lambda$, thus, with high probability,

$$|P - AB|_2 \leq 2nM^2(m+p)^{1/k}\sqrt{\frac{m+p}{s}}$$
$$+2nM^2\sqrt{32\ln 4(m+p)} + 128nM^2$$

Setting e.g. $k = \ln(m+p)$ and observing that for $m + p > x_0$ (for some x_0) the first term of the right hand side dominates, the above inequality becomes

$$|P - AB|_2 \leq c \cdot nM^2\sqrt{\frac{m+p}{s}}$$

for some constant $c > 2e$. Our assumption on s now becomes $s \leq \frac{m+p}{4\ln^6(m+p)}$ (the 11^6 term has disappeared).