Pass-Efficient Algorithms for Approximating Large Matrices

Petros Drineas *

1 Introduction

We are interested in developing and analyzing fast Monte Carlo algorithms for performing useful computations on large matrices. We consider new methods for common problems such as matrix multiplication, the Singular Value Decomposition (SVD), and the computation of a compressed approximate decomposition of a large matrix. Since such computations generally require time which is superlinear in the number of nonzero elements of the matrix, we expect our algorithms to be useful in many applications where data sets are modeled by matrices and are extremely large. In all these cases, we assume that the input matrices are prohibitively large to store in Random Access Memory (RAM) and thus that only external memory storage is possible. Our algorithms will be allowed to read the matrices a few, e.g., one or two or three, times and keep a small randomly-chosen and rapidly-computable "sketch" of the matrices in RAM; computations will then be performed on this "sketch". We will work within the framework of the Pass-Efficient computational model, in which the scarce computational resources are the number of passes over the data, the additional RAM space required, and the additional time required [8, 10].

Recent interest in computing with massive data sets has led to the development of computational models in which the usual notions of time-efficiency and space-efficiency have been modified [3, 8, 13, 14, 15]. In the applications that motivate these data-streaming models the data sets are much too large to fit into main memory. Thus, they are either not stored or are stored in a secondary storage device which may be read sequentially as a data stream but for which random access is very expensive. Typically, algorithms that compute on a data stream examine the data stream, keep a small "sketch" of the data, and perform computations on the sketch. Thus, these algorithms are usually randomized and approximate, and their performance is evaluated by considering resources such as the time to process an item in the data stream, the number of passes over the data, the additional workspace and additional time required, and the quality of the approximations returned. The motivation for our particular "pass-efficient" approach is that in modern computers the amount of disk storage (external memory) has increased enormously, while RAM and computing speeds have increased, yet at a substantially slower pace. Thus, we have the ability to store large amounts of data, but not in RAM, and we do not have the computational ability to process these data with algorithms that require superlinear time.

2 The Matrix Multiplication Algorithm

We present a simple, novel algorithm for the Matrix Multiplication Problem. Suppose A and B (which are $m \times n$ and $n \times p$ respectively) are the two input matrices. In our main algorithm, we perform $c = O(1)$ independent trials, where in each trial we randomly sample an element of $\{1, 2, \ldots n\}$ with an appropriate probability distribution $\mathcal{P}$ on $\{1, 2, \ldots n\}$. We form a $m \times c$ matrix C consisting of the sampled columns of A, each scaled appropriately, and we form a $c \times n$ matrix R using the same rows of B, again scaled appropriately. The choice of $\mathcal{P}$ and the column and row scaling are crucial features of the algorithm. When these are chosen judiciously, we prove that CR is a good approximation to AB; more precisely, we show that, with high probability,

$$\|AB - CR\|_F \in O(\|A\|_F \|B\|_F / \sqrt{c}),$$

*Department of Computer Science, Rensselaer Polytechnic Institute, Troy, New York 12180.

where $\|\cdot\|_F$ denotes the Frobenius norm, i.e., $\|A\|_F^2 = \sum_{i,j} A_{ij}^2$. This algorithm can be implemented without storing the matrices A and B in RAM, provided it can make two passes over the matrices stored in external memory and use $O(m+p)$ additional RAM memory to construct C and R.

3 The CUR approximation algorithm

We subsequently present an algorithm which, when given an $m \times n$ matrix A, computes approximations to A which are the product of three smaller matrices, C, U, and R, each of which may be computed rapidly. Let $A' = CUR$ be the computed approximate decomposition; our algorithm has provable bounds for the error matrix $A - A'$. The CUR algorithm chooses $c = O(1)$ columns of A and $r = O(1)$ rows of A randomly; if the $m \times c$ matrix C consists of those c columns of A (after appropriate rescaling) and the $r \times n$ matrix R consists of those r rows of A (also after appropriate rescaling) then the $c \times r$ matrix U may be calculated from C and R. For any matrix X, let $\|X\|_F$ and $\|X\|_2$ denote its Frobenius norm and its spectral norm, respectively. It is proven that

$$\|A - A'\|_\xi \leq \min_{D:\mathrm{rank}(D)\leq k} \|A - D\|_\xi + poly(k, 1/c) \|A\|_F$$

holds in expectation and with high probability for both $\xi = 2, F$ and for all $k = 1, \ldots, rank(A)$; thus by appropriate choice of k

$$\|A - A'\|_2 \leq \epsilon \|A\|_F$$

also holds in expectation and with high probability. This algorithm may be implemented without storing the matrix A in Random Access Memory (RAM), provided it can make two passes over the matrix stored in external memory and use $O(m+n)$ additional RAM memory. To achieve an additional error (beyond the best rank k approximation) that is at most $\epsilon \|A\|_F$, the CUR algorithm takes time which is a low-degree polynomial in $\max(m, n)$, k, $1/\epsilon$, and $1/\delta$. The proofs for the error bounds make important use of matrix perturbation theory and previous work on approximating matrix multiplication and computing low-rank approximations to a matrix. The probability distribution over columns and rows and the rescaling are crucial features of the algorithms and must be chosen judiciously.

4 A PTAS for the weighted Max-Cut Problem on dense graphs

Recent work in the development and analysis of randomized approximation algorithms for NP-hard problems has involved approximating the solution to a problem by the solution to an induced subproblem of constant size, where the subproblem is constructed by sampling elements of the original problem uniformly at random. In light of interest in problems with a heterogeneous structure, for which uniform sampling might be expected to yield suboptimal results, we investigate the use of nonuniform sampling probabilities. We show that by judicious choice of sampling probabilities and a variant of the CUR approximation algorithm, one can obtain error bounds that are superior to the ones obtained by uniform sampling for weighted versions of the Max-Cut problem, for certain regimes of the error parameter ϵ. Of particular interest is one of our techniques: we develop a method to approximate the feasibility of a large linear program by a nonuniformly randomly chosen subprogram; for more details see [9].

5 Related Work

In other related work, Achlioptas and McSherry have also computed succinctly-described matrix approximations using somewhat different sampling techniques [1, 2]. Also included in [1, 2] is a comparison of their methods with those of [8].

Recent work has focused on developing new techniques for proving lower bounds on the number of queries a sampling algorithm is required to perform in order to approximate a given function accurately with a low probability or error [4].

Acknowledgements: The aforementioned results emerged from joint work with Ravi Kannan (Computer Science Department, Yale University) and Michael W. Mahoney (Math Department, Yale University); more details can be found in [5, 6, 7, 8, 9, 10, 11, 12].

References

[1] D. Achlioptas and F. McSherry. Fast computation of low rank matrix approximations. *submitted*.

[2] D. Achlioptas and F. McSherry. Fast computation of low rank matrix approximations. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 611–618, 2001.

[3] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 20–29, 1996.

[4] Z. Bar-Yossef. Sampling lower bounds via information theory. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 335–344, 2003.

[5] P. Drineas. *Randomized Algorithms for Matrix Operations*. PhD thesis, Yale University, 2003.

[6] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering in large graphs and matrices. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 291–299, 1999.

[7] P. Drineas and R. Kannan. Fast Monte-Carlo algorithms for approximate matrix multiplication. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 452–459, 2001.

[8] P. Drineas and R. Kannan. Pass efficient algorithms for approximating large matrices. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 2003.

[9] P. Drineas, R. Kannan, and M.W. Mahoney. Sampling sub-problems of Max-Cut problems and approximation algorithms. *manuscript*.

[10] P. Drineas, R. Kannan, and M.W. Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. Technical Report YALEU/DCS/TR-1269, Yale University Department of Computer Science, New Haven, CT, February 2004.

[11] P. Drineas, R. Kannan, and M.W. Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. Technical Report YALEU/DCS/TR-1270, Yale University Department of Computer Science, New Haven, CT, February 2004.

[12] P. Drineas, R. Kannan, and M.W. Mahoney. Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition. Technical Report YALEU/DCS/TR-1271, Yale University Department of Computer Science, New Haven, CT, February 2004.

[13] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L^1-difference algorithm for massive data sets. In *Proceedings of the 40th Annual IEEE Symposium on the Foundations of Computer Science*, pages 501–511, 1999.

[14] M.R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. Technical Report 1998-011, Digital Systems Research Center, Palo Alto, CA, May 1998.

[15] J.I. Munro and M.S. Paterson. Selection and sorting with limited storage. In *Proceedings of the 19th Annual IEEE Symposium on Foundations of Computer Science*, pages 253–258, 1978.