A Fast Random Sampling Algorithm for Sparsifying Matrices

Sanjeev Arora* Elad Hazan* Satyen Kale*
Computer Science Department, Princeton University
35 Olden Street, Princeton, NJ 08540
{arora, ehazan, satyen}@cs.princeton.edu

May 25, 2006

Abstract

We describe a simple random-sampling based procedure for producing sparse matrix approximations. Our procedure and analysis are extremely simple: the analysis uses nothing more than the Chernoff-Hoeffding bounds. Despite the simplicity, the approximation is comparable and sometimes better than previous work.

Our algorithm computes the sparse matrix approximation in a single pass over the data. Further, most of the entries in the output matrix are quantized, and can be succinctly represented by a bit vector, thus leading to much savings in space.

1 Introduction

Eigenvector computations are ubiquitous in numerous algorithmic tasks: a few applications include clustering in high dimensional data, principal component analysis, spectral graph partitioning, semidefinite programming, and Google's PageRank algorithm. Because of the central importance of eigenvector computations, this problem has been very well studied by numerical analysts.

In practical applications one frequently needs to compute eigenvectors of matrices arising from massive data sets such as web corpora, images, or video. Any superlinear computation quickly becomes infeasible as the matrix size becomes large. If approximate eigenvectors are allowed, then one can use the power method and the Lanczos method [TB97] which are very efficient in practice. These two methods spend the bulk of their processing time in computing matrix-vector products. Computing a matrix-vector product takes time proportional to the number of non-zero entries in the matrix (i.e. the *sparsity* of the matrix), and suggests that the eigenvector computation could be sped up by sparsifying the matrix first. This involves computing a different matrix that has fewer non-zero entries than the original, yet remains close to it by some metric. Section 2 makes these notions precise and describes how such a sparse matrix approximation can be used as a proxy for the original in the eigenvector computation.

Frieze, Kannan and Vempala [FKV04] considered the problem of efficiently computing low-rank approximations to matrices, and gave an algorithm to perform the task via random sampling of the columns of the input matrix with carefully chosen probabilities. Since many columns are discarded in the random sampling process, the algorithm can be interpreted as computing a sparse representation of the input matrix (albeit only for the specific application of computing low-rank

*Supported by Sanjeev Arora's NSF grants MSPA-MCS 0528414, CCF 0514993, ITR 0205594.

approximations). This work was later refined and extended by Drineas *et al* [DFK$^+$04] and Drineas and Kannan [DK03].

Achlioptas and McSherry [AM01] gave an algorithm that sparsifies the input matrix via random sampling of the entries rather than the columns. They applied their sparsification algorithm to the problem of computing low-rank approximations to matrices: the idea was to simply use the sparsified matrix in the orthogonal or Lanczos iteration algorithms for computing the approximations. They gave precise error estimates for the low-rank approximation in terms of the sparsification quality. Their best algorithm has better performance than [DFK$^+$04, DK03] in minimizing the ℓ_2 norm of the difference matrix, and comparable performance in the Frobenius norm. In addition, they require only one pass over the input matrix instead of the two passes needed for previous work. Furthermore, [AM01] also describe a *quantization* algorithm: this algorithm transforms all non-zero entries of the input matrix into entries with the same magnitude, and so the output matrix can be succinctly represented by a bit vector corresponding to the sign of the entries.

The purpose of this note is to give a new and simple sparsification algorithm that has comparable performance to the algorithm of Achlioptas and McSherry, and is better in situations when the allowed approximation error is small. This is because the dependence on the approximation error ϵ is $\frac{1}{\epsilon}$ for our algorithm vs. $\frac{1}{\epsilon^2}$ for [AM01], so our algorithm scales better when the error that can be tolerated goes down. Another advantage is that it runs in a single pass over the input matrix and produces quantized entries directly (without the need for an extra quantization step like [AM01]). The analysis is particularly simple: all that is needed are the well-known Chernoff-Hoeffding bounds. This algorithm arose in applications in fast semidefinite programming [AHK05], where our algorithm gives better performance than that of [AM01]. In this paper, we abstract out the algorithm and refine the details that were hidden in the specific applications of [AHK05].

2 Preliminaries

Given an input matrix A, an ϵ-approximation for A is a matrix $\tilde{A}$ such that $\|A - \tilde{A}\|_2 \leq \epsilon$. Here, $\|A\|_2 := \max_{\|x\|_2=1} |Ax|$ is the ℓ_2 norm of A. For symmetric matrices A, $\|A\|_2$ is the magnitude of the largest eigenvalue in absolute value. We assume without loss of generality that the input matrix A is a symmetric, $n \times n$ real matrix. This is because given an arbitrary $n \times m$ real matrix B, we can instead consider the symmetric $(n+m) \times (n+m)$ matrix

$$A = \begin{pmatrix} 0 & B \\ B^\top & 0 \end{pmatrix}$$

which has the same ℓ_2 norm as B, and whose ϵ-approximation gives an ϵ-approximation for B in the obvious way.

Now, we will make precise what it means to compute an approximate eigenvector. For a matrix A, let v be a unit eigenvector corresponding to the largest eigenvalue, so that the largest eigenvalue of A is $v^\top Av$. A unit vector u will be called an ϵ-approximate largest eigenvector of A if $u^\top Au \geq v^\top Av - \epsilon$. Let $\tilde{A}$ be an ϵ-approximation of A, and let u be an arbitrary unit vector. Then we have

$$|u^\top (A - \tilde{A})u| \;\leq\; \|A - \tilde{A}\|_2 \;=\; \epsilon.$$

Let u be the unit eigenvector corresponding to the largest eigenvalue of $\tilde{A}$. Then

$$v^\top Av \;\leq\; v^\top \tilde{A}v + \epsilon \;\leq\; u^\top \tilde{A}u + \epsilon.$$

which implies that u is an ϵ-approximate largest eigenvector of A.

3 Algorithm and Comparison of Results

The procedure SPARSIFY in Figure 1 computes a sparse, $O(\epsilon)$-approximation to an input matrix A. Theorem 1 gives the performance guarantee for the procedure SPARSIFY, and is proved in Section 4.

Procedure SPARSIFY(A, ϵ)
for each $i, j \in [n]$ **do**
if $|A_{ij}| > \frac{\epsilon}{\sqrt{n}}$ **then**
$\tilde{A}_{ij} = A_{ij}$
else

$$\tilde{A}_{ij} = \begin{cases} \operatorname{sgn}(A_{ij}) \cdot \frac{\varepsilon}{\sqrt{n}} & \text{with probability } p_{ij} = \frac{\sqrt{n}|A_{ij}|}{\epsilon} \\ 0 & \text{with probability } 1 - p_{ij} \end{cases}$$

return $\tilde{A}$

Figure 1: Procedure SPARSIFY

Theorem 1 *Let $A \in \mathbb{R}^{n \times n}$ be a matrix with N non-zero entries and let $S = \sum_{ij} |A_{ij}|$. Let $\epsilon > 0$ be a given error parameter. Then the procedure* SPARSIFY *runs in $O(N)$ time (a single pass over the input matrix) and produces a matrix $\tilde{A}$ such that:*

1. *With probability at least $1 - \exp(-\Omega(\frac{\sqrt{n}S}{\delta}))$, $\tilde{A}$ has $O(\frac{\sqrt{n}S}{\delta})$ non-zero entries, and*
2. *With probability at least $1 - \exp(-\Omega(n))$, we have $\|A - \tilde{A}\|_2 \leq O(\epsilon)$.*

Now, we give a comparison of our results with previous work. Achlioptas and McSherry [AM01] have a very detailed comparison of the use of the sparsification algorithm with the algorithms of [FKV04, DFK+04, DK03] for the task of computing low-rank matrix approximations, so we refer the interested reader to [AM01] for this specific application. In this section, we only compare the algorithm of [AM01] to ours for the task of sparsifying an input matrix.

The strongest result of [AM01] is a random sampling algorithm, that, in one pass over the input matrix A, computes a matrix $\tilde{A}$ such that with probability at least $1 - 1/n$, we have $\|A - \tilde{A}\|_2 \leq \epsilon$, and which retains an expected $\tilde{O}(\frac{n}{\epsilon^2} \sum_{ij} A_{ij}^2 + n)$ non-zero entries.

In comparison, our algorithm computes, in one pass over the input matrix A, a matrix $\tilde{A}$ such that with probability at least $1 - \exp(-\Omega(n))$, we have $\|A - \tilde{A}\|_2 \leq \epsilon$, and which retains an expected $\tilde{O}(\frac{\sqrt{n}}{\epsilon} \sum_{ij} |A_{ij}|)$ non-zero entries.

Thus, our algorithm has exponentially lower failure probability and better dependence on the error parameter ϵ (linear, rather than quadratic), and on the input matrix order n.

Our algorithm also has the advantage that barring a few large entries, the sampled entries are all quantized: since their magnitude is always $\frac{\epsilon}{\sqrt{n}}$, they can be represented very succinctly by just their sign. This can result in considerable savings in the space needed to store the sampled matrix. [AM01] also have an algorithm which can quantize a matrix, but it does not lead to any sparsification by itself. It can be applied to the sparsified matrix rather than the original one to obtain some amount of quantization. However, the error bound of the quantization process depends on the largest entry in the sparsified matrix, which curtails the benefits of quantization.

To elucidate how the choice of the error parameter affects the performance of the two algorithms, we consider two cases. In the first case, the error ϵ is of the order of the ℓ_2 norm of A, viz. $\epsilon = \delta\|A\|_2$. This situation arises in applications such as solving semidefinite programs efficiently [AHK05]. In the second case, the error ϵ is of the order of the Frobenius norm of A, viz. $\epsilon = \delta\|A\|_F = \delta\sqrt{\sum_{ij} A_{ij}^2}$. This situation arises in applications such as computing low-rank approximations to matrices [FKV04, DFK$^+$04, DK03, AM01].

The first error is typically much smaller than the second, so our algorithm can be expected to perform better in the first case, and Achlioptas and McSherry's in the second. To illustrate this, consider the two matrices $I + \frac{1}{n}J$ where I is the identity matrix, and J is the all 1's matrix; and the matrix H which is the Hadamard matrix of order n (assuming it exists). The following table summarizes the sparsification achieved by the algorithms:

Algorithm	Matrix	$\epsilon = \delta\|A\|_2$	$\epsilon = \delta\|A\|_F$
This paper	$I + \frac{1}{n}J$	$O\left(\frac{n^{1.5}}{\delta}\right)$	$O\left(\frac{n}{\delta}\right)$
[AM01]	$I + \frac{1}{n}J$	$O\left(\frac{n^2}{\delta^2}\right)$	$O\left(\frac{n}{\delta^2}\right)$
This paper	H	no sparsification	$O\left(\frac{n^{1.5}}{\delta}\right)$
[AM01]	H	no sparsification	$O\left(\frac{n}{\delta^2}\right)$

In summary, the our algorithm is better in some situations than that of [AM01] and worse in others. Exactly which algorithm to use in a given situation depends on the input parameters. A general guideline is that our algorithm is preferable when one needs a high degree of accuracy.

4 Analysis

Lemma 1 *With probability at least* $1 - \exp(-\Omega(\frac{\sqrt{n}S}{\epsilon}))$*, the matrix* $\tilde{A}$ *contains at most* $O(\frac{\sqrt{n}S}{\epsilon})$ *non-zero entries.*

PROOF: Since $\sum_{ij} |A_{ij}| = S$, the number of A_{ij} that are larger than $\frac{\epsilon}{\sqrt{n}}$ is at most $\frac{\sqrt{n}S}{\epsilon}$. Consider now all non-zero A_{ij} entries that are smaller than $\frac{\epsilon}{\sqrt{n}}$.

The Chernoff bound [MR95] asserts that if $X_1, X_2, \ldots, X_n$ are indicator random variables and $X = \sum_i X_i$ with $\mathbb{E}[X] = \mu$, then

$$\mathbf{Pr}[X > (1+\delta)\mu] < \left[\frac{e^\delta}{(1+\delta)^{1+\delta}}\right]^\mu$$

In our case, we set up indicator random variables X_{ij} which are 0 or 1 depending on whether $\tilde{A}_{ij} = 0$ or not. Then $X = \sum_{ij} X_{ij}$ is the number of non-zero entries of $\tilde{A}$, and $\mathbb{E}[X] = \sum_{ij} p_{ij} = \frac{\sqrt{n}S}{\epsilon}$. The claim follows by using the Chernoff bound with $(1+\delta) = e$. □

Next, define $M = A - \tilde{A}$. We will show that for all unit vectors x, $|x^\top M x| \leq O(\epsilon)$ which implies $\|A - \tilde{A}\|_2 \leq O(\epsilon)$. Notice that $M_{ij} = 0$ for all coordinates i, j such that $|A_{ij}| \geq \frac{\epsilon}{\sqrt{n}}$. For the rest of the coordinates, since $\mathbb{E}[\tilde{A}_{ij}] = \mathrm{sgn}(A_{ij}) \cdot \frac{\epsilon}{\sqrt{n}} \times \frac{\sqrt{n}|A_{ij}|}{\epsilon} = A_{ij}$, we conclude that $\mathbb{E}[M_{ij}] = 0$. We will now consider a $\frac{\epsilon_0}{\sqrt{n}}$-grid on the unit sphere (ϵ_0 is set to some constant, say $\frac{1}{2}$),

$$T = \left\{ x : \; x \in \frac{\epsilon_0}{\sqrt{n}}\mathbb{Z}^n, \; \|x\|_2 \leq 1 \right\}.$$

Feige and Ofek [FO05] give a bound on the size of T and show that it suffices to consider only vectors in T (we reprove this in Appendix A):

Lemma 2 *The size of $|T|$ is at most $\exp(cn)$ for $c = (\frac{1}{\epsilon_0} + 2)$. If for every $x, y \in T$ we have $|x^\top My| \leq \epsilon$, then for every unit vector x, we have $|x^\top Mx| \leq \frac{\epsilon}{(1-\epsilon_0)^2}$.*

Let $x, y \in T$. Since $\mathbb{E}[M_{ij}] = 0$, we conclude that $\mathbb{E}[x^\top My] = 0$. We now a prove strong concentration bound:

Lemma 3 *With probability at least $1 - \exp(-\Omega(n))$, for every $x, y \in T$ it holds that $|x^\top My| \leq c\epsilon$.*

PROOF: We use the following bound from Hoeffding's original paper [Hoe63]: let $X_1, ..., X_n$ be independent random variables, such that X_i takes values in the range $[a_i, b_i]$. Let $X = \sum_i X_i$, and $\mathbb{E}[X] = \mu$. Then for any $t > 0$

$$\Pr[|X - \mu| \geq t] \;\leq\; 2\exp\left(-\frac{2t^2}{\sum_i (b_i - a_i)^2}\right).$$

Consider the random variables $Z_{ij} = M_{ij}x_iy_j$, then $x^\top My = \sum_{ij} M_{ij}x_iy_j = \sum_{ij} Z_{ij}$. Since $\tilde{A}_{ij}$ is either $\mathrm{sgn}(A_{ij}) \cdot \frac{\epsilon}{\sqrt{n}}$ or 0, the squared range of M_{ij} is $\frac{\epsilon^2}{n}$. Thus, the sum of squared ranges for the variables Z_{ij} is $\leq \sum_{ij} \frac{\epsilon^2}{n} x_i^2 y_j^2 = \frac{\epsilon^2}{n} \sum_i x_i^2 \sum_j y_j^2 \leq \frac{\epsilon^2}{n}$. Since $E[Z_{ij}] = 0$, by the Hoeffding bound we have:

$$\Pr[|x^\top My| \;\geq\; c\epsilon] \;\leq\; 2\exp\left(-\frac{2c^2\epsilon^2}{\frac{\epsilon^2}{n}}\right) \;=\; 2\exp(-2c^2 n)$$

Since there are $\exp(2cn)$ pairs of vectors $x, y \in T$, the union bound implies that with probability at least $1 - \exp(-\Omega(n))$, for all vectors $x, y \in T$, we have $|x^\top My| \leq c\epsilon$. □

5 Conclusions

In this paper, we presented a fast and simple random sampling algorithm to sparsify matrices, with comparable performance guarantees to previous work. The analysis of the algorithm is also fairly easy, relying only on the well-known Chernoff-Hoeffding bounds. The algorithm has better dependence on the error parameter than previous work, which makes it preferable when low error is desired.

However, its dependence on the input matrix size may be worse than previous algorithms in situations where all entries are roughly the same magnitude. This suggests that in practice, a hybrid algorithm combining ours with that of Achlioptas and McSherry may be able to strike a better balance between the dependence on the error parameter and input size.

References

[AHK05] Sanjeev Arora, Elad Hazan, and Satyen Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *46th FOCS*, pages 339–348, 2005.

[AM01] Dimitris Achlioptas and Frank McSherry. Fast computation of low rank matrix approximations. In *32nd STOC*, pages 611–618, 2001.

[DFK+04] Petros Drineas, Alan M. Frieze, Ravi Kannan, Santosh Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, 56(1-3):9–33, 2004.

[DK03] Petros Drineas and Ravi Kannan. Pass efficient algorithms for approximating large matrices. In *SODA*, pages 223–232, 2003.

[FKV04] Alan M. Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004.

[FO05] U. Feige and E. Ofek. Spectral techniques applied to sparse random graphs. *Random Structures and Algorithms*, 27(2):251–275, September 2005.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge Univ. Press, 1995.

[TB97] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997.

A Discretization

In this section, we prove Lemma 2. We restate it here for convenience:

Lemma 4 *Let* $T = \left\{x : \ x \in \frac{\epsilon_0}{\sqrt{n}}\mathbb{Z}^n, \ \|x\|_2 \leq 1\right\}$. *The size of* T *is at most* $\exp(cn)$ *for* $c = (\frac{1}{\epsilon_0} + 2)$. *If for every* $x, y \in T$ *we have* $|x^\top My| \leq \epsilon$ *then for every unit vector* x, *we have* $|x^\top Mx| \leq \frac{\epsilon}{(1-\epsilon_0)^2}$.

PROOF: Map every point in $x \in T$ in a one-to-one correspondence with a n-dimensional hypercube of side length $\frac{\epsilon_0}{\sqrt{n}}$ on the grid: $x \mapsto C_x = \left\{x + u : \ u \geq \vec{0}, \ \|u\|_\infty \leq \frac{\epsilon_0}{\sqrt{n}}\right\}$. The maximum length of any vector in C_x is bounded by $\|x\| + \epsilon_0 \leq 1 + \epsilon_0$, and thus the union of these cubes is contained in the n-dimensional ball B of radius $(1 + \epsilon_0)$. We conclude:

$$|T| \times \left(\frac{\epsilon_0}{\sqrt{n}}\right)^n \ = \ \sum_{x \in T} \mathbf{Vol}(C_x) \ \leq \ \mathbf{Vol}(B) \ = \ \frac{\pi^{n/2}}{\Gamma(n/2+1)}(1+\epsilon_0)^n.$$

And so:

$$|T| \ \leq \ \frac{\pi^{n/2}}{\Gamma(n/2+1)}\left(\frac{(1+\epsilon_0)\sqrt{n}}{\epsilon_0}\right)^n \ \leq \ \exp\left(\left(\frac{1}{\epsilon_0} + 2\right)n\right).$$

Next, given any unit vector, x, let $y = (1 - \epsilon_0)x$. By "rounding down" the coordinates of y to the nearest multiple of $\frac{\epsilon_0}{\sqrt{n}}$, we get a grid point z such that $y \in C_z$. Thus, the maximum length of any vertex of C_z is bounded by $\|y\| + \epsilon_0 = 1$, so all vertices of C_z are grid points in T. Express y as a convex combination of the vertices v_i of C_z; viz. $y = \sum_i \alpha_i v_i$ with $\alpha_i \geq 0$ and $\sum_i \alpha_i = 1$. Then we have

$$|y^\top My| \ = \ |(\sum_i \alpha_i v_i)^\top M(\sum_i \alpha_i v_i)| \ \leq \ \sum_{i,j} \alpha_i\alpha_j |v_i^\top M v_j| \ \leq \sum_{i,j} \alpha_i\alpha_j\epsilon \ = \ \epsilon.$$

The second inequality above follows because we assumed that for all $x, y \in T$, $|x^\top My| \leq \epsilon$. Finally, since $y = (1 - \epsilon_0)x$, we have

$$|x^\top Mx| \ = \ \frac{|y^\top My|}{(1-\epsilon_0)^2} \ \leq \ \frac{\epsilon}{(1-\epsilon_0)^2}.$$

□