SIAM J. SCI. COMPUT.
Vol. 33, No. 4, pp. 1689–1706

IMPORTANCE SAMPLING FOR A MONTE CARLO MATRIX MULTIPLICATION ALGORITHM, WITH APPLICATION TO INFORMATION RETRIEVAL*

SYLVESTER ERIKSSON-BIQUE†, MARY SOLBRIG‡, MICHAEL STEFANELLI§, SARAH WARKENTIN¶, RALPH ABBEY‖, AND ILSE C. F. IPSEN‖

Abstract. We perform importance sampling for a randomized matrix multiplication algorithm by Drineas, Kannan, and Mahoney and derive probabilities that minimize the expected value (with regard to the distributions of the matrix elements) of the variance. We compare these optimized probabilities with uniform probabilities and derive conditions under which the actual variance of the optimized probabilities is lower. Numerical experiments with query matching in information retrieval applications illustrate that the optimized probabilities produce more accurate matchings than the uniform probabilities and that they can also be computed efficiently.

Key words. randomized algorithm, expected value, variance, term-by-document matrix, query matching

AMS subject classifications. 68W20, 68Q87, 62P30, 15B48, 15A99

DOI. 10.1137/10080659X

1. Introduction.

Randomized algorithms have shown potential for a variety of matrix computations, including norms and inner products [6, 13, 26, 27], matrix products [1, 6, 9], as well as matrix decompositions and low rank approximations [4, 10, 12, 14, 19]. An excellent survey of and introduction to randomized algorithms for matrix computations is given in [14]. The reason for the growing popularity of randomized algorithms is the increasingly common occurrence of massive data sets that are too large for traditional deterministic algorithms.

The randomized algorithm under consideration is a Monte Carlo algorithm for matrix multiplication by Drineas and Kannan [8] and Drineas, Kannan, and Mahoney [9]. Monte Carlo algorithms approximate a desired value by means of repeated sampling [17, section 10], [18, section 16]. In the theoretical computer science community, a *Monte Carlo algorithm* refers more specifically to a randomized algorithm "that may fail or return an incorrect answer" [17, p. 57] but whose time complexity often does not depend on the particular sampling. This is in contrast to a *Las Vegas algorithm*, which always returns the correct answer but whose time complexity is not deterministic.

Among randomized algorithms for inner products and matrix multiplication [1, 6, 8, 9, 13, 14, 19, 26, 27], the Monte Carlo algorithms [8, 9] are not necessarily the ones with the best asymptotic accuracy. We chose them simply because they are straightforward to implement and adapt to the requirements of query matching problems

*Submitted to the journal's Methods and Algorithms for Scientific Computing section August 25, 2010; accepted for publication (in revised form) April 26, 2011; published electronically July 26, 2011. This work was supported in part by NSF grant DMS055271 and NSA grants H98230-08-1-0094 and H9823-10-1-0252.

http://www.siam.org/journals/sisc/33-4/80659.html

†University of Helsinki, Helsinki FI-00014, Finland (sylvester.eriksson-bique@helsinki.fi).

‡Reed College, Portland, OR 97202 (solbrigm@reed.edu).

§College of New Jersey, Ewing, NJ 08628 (stefane3@tcnj.edu).

¶Harvey Mudd College, Claremont, CA 91711 (swarkentin@hmc.edu).

‖Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205 (rwabbey@ncsu.edu, ipsen@math.ncsu.edu).

and because they allowed us to incorporate information about the distribution of the inputs and give a realistic analysis of the accuracy.

The Monte Carlo algorithms in [8, 9] approximate the inner products $a^T b$ between two vectors a and b by sampling a few elements from a and the corresponding elements from b, according to a probability vector p supplied by the user. The resulting approximation X is an unbiased estimator; that is, X is a random variable whose expected value is equal to the desired inner product $a^T b$, i.e., $\mathbf{E}[X] = a^T b$.

However, just because an algorithm is an unbiased estimator does not imply it is accurate, since the output X from a single run of the algorithm can be far from the expected value $\mathbf{E}[X]$. The deviation of a random variable X from its expected value is measured by the variance,

$$\mathbf{Var}[X] = \mathbf{E}\left[(X - \mathbf{E}[X])^2\right].$$

A simple way to get from the variance to an actual error bound is Chebyshev's inequality:

$$\text{The probability that} \quad |X - \mathbf{E}[X]| \leq \tau \quad \text{is at least } 1 - \mathbf{Var}[X]/\tau^2.$$

In other words,

$$|X - \mathbf{E}[X]| \leq \sqrt{\mathbf{Var}[X]/\delta} \quad \text{with probability at least } 1 - \delta, \tag{1.1}$$

which means that the difference between a random variable and its expected value can be estimated by the square root of the variance. The derivation of sampling probabilities p for the purpose of reducing variance is called *importance sampling* [18, 22, 21]. The randomized matrix multiplication algorithm by Drineas and Kannan [8] and Drineas, Kannan, and Mahoney [9] can be interpreted as a Monte Carlo algorithm with importance sampling (section 2.2).

Our approach to importance sampling is motivated by an application in information retrieval (section 4), where a query vector is "matched" against documents in a collection. This matching process amounts to computing a sequence of inner products between the query vector and the documents in the collection. In our approach, we approximate this sequence of inner products by a randomized matrix multiplication algorithm. As soon as the query arrives, the sampling probabilities need to be computed fast. To this end, we derive probabilities that minimize the expected value—with regard to the distributions of the matrix elements—of the variance (section 2.3). This works for continuous distributions (section 2.5) as well as discrete distributions (section 4.1).

We establish theoretical conditions (section 2.4) under which our optimized probabilities have a variance that is smaller than that of uniform probabilities. We illustrate that the optimized probabilities can also be more efficient in practice. In document matchings with Reuters and Wikipedia data sets (sections 4.2 and 4.3), where one is interested only in ranking the matches, the optimized probabilities produce better rankings. Furthermore, the optimized probabilities can be computed fast if the document distributions are available offline. The online computation requires only a single probability vector per query and does not depend on the number of documents in the collection.

Overview. Importance sampling is discussed for inner products in section 2 and for matrix multiplication in section 3. The information retrieval application is presented in section 4. We end with conclusions in section 5.

2. Inner products. We interpret a randomized algorithm for computing inner products from [8, 9] in the context of Monte Carlo algorithms with importance sampling. Then we introduce an approach to importance sampling that minimizes the expected value of the variance (with regard to the distribution of the vector elements).

We present the randomized algorithm in section 2.1 and its interpretation in terms of importance sampling in section 2.2. We derive the probabilities for minimizing the expected variance in section 2.3 and compare the actual variance to that of uniform probabilities in section 2.4. In section 2.5 we present numerical experiments on vectors whose elements come from continuous distributions, to illustrate the higher accuracy of our "optimized" probabilities.

2.1. The algorithm. The randomized algorithm for computing the inner product is displayed as Algorithm 1. It is the BasicMatrixMultiplication algorithm [9, Figure 2] and the algorithm in [8, section 5] specialized to computing inner products.

Given real $n \times 1$ vectors a and b, Algorithm 1 approximates $a^T b$ by sampling c elements from a and the corresponding elements of b according to given probabilities[1] p_i. The algorithm uses sampling with replacement, which means that an element can be sampled repeatedly. An intuitive, but not necessarily efficient, way to do this is presented as Algorithm 2. Sampling with replacement is easier to implement and to analyze than sampling without replacement, and our experiments did not not show a significant difference in accuracy.

Algorithm 1. Inner Product Algorithm [8, 9]

Input: real vectors $a = \begin{pmatrix} a_1 & \dots & a_n \end{pmatrix}^T$, $b = \begin{pmatrix} b_1 & \dots & b_n \end{pmatrix}^T$
probabilities $p = \begin{pmatrix} p_1 & \dots & p_n \end{pmatrix}^T$ with $p_i > 0$ and $\sum_{i=1}^n p_i = 1$
integer c, where $1 \le c \le n$

Output: Approximation X to $a^T b$ from c elements of a and b
sampled according to probabilities p_i

```
X = 0
for t = 1 : c do
   Sample i_t from {1, ..., n} with probability p_{i_t}
      independently and with replacement
   X = X + a_{i_t} b_{i_t} / (c p_{i_t})
end for
return X
```

Drineas, Kannan, and Mahoney [9, Lemma 3] and Drineas and Kannan [8, section 5] derive the following expressions for the expected value and variance of the output of Algorithm 1:

$$\mathbf{E}[X] = a^T b, \qquad \mathbf{Var}[X] = \frac{1}{c}\left(\sum_{i=1}^{n} \frac{a_i^2 b_i^2}{p_i} - (a^T b)^2\right). \tag{2.1}$$

This shows that the expected value of X is equal to the desired inner product, so that the output of Algorithm 1 is an unbiased estimator for $a^T b$.

[1] We assume that the probabilities are positive, but we could modify Algorithm 1 to allow zero probabilities for zero products $a_i b_i$. This would simply amount to removing elements associated with zero products from a and b and then applying Algorithm 1 with positive probabilities to the shorter vectors.

In their papers on Fourier transforms, Gilbert et al. [13] and Zou et al. [27] present a randomized inner product algorithm that essentially computes the median of several runs of Algorithm 1. However, in our experiments we could not discern a definite increase in accuracy with this approach and thus did not pursue it.

Implementation of sampling. Sampling with replacement is easy if the probabilities are uniform, i.e., $p_i = 1/n$, $1 \le i \le n$. In contrast, sampling with nonuniform probabilities is not efficient for a single inner product computation, because it can be too expensive. However, in the context of matrix multiplication, nonuniform sampling can be efficient; see [8, 9] and section 4.

In order to illustrate how sampling with replacement can be implemented computationally, we present the simple and intuitive but inefficient method of *inversion by sequential search* [7, section III.2.1]; see Algorithm 2. From the given probabilities p_i, define the sums $S_t \equiv \sum_{1 \le i \le t} p_i$. Pick a uniform $[0, 1]$ random variable U, and let m be the integer with $S_{m-1} < U \le S_m$. Then the probability that $m = k$ equals $S_k - S_{k-1} = p_k$.

ALGORITHM 2. Sampling an Index with Replacement [7]

Input: Probabilities $p = \begin{pmatrix} p_1 & \dots & p_n \end{pmatrix}^T$ with $p_i > 0$ and $\sum_{i=1}^n p_i = 1$
Output: Index k from $\{1, \dots, n\}$ sampled independently and with replacement according to probabilities p_i

$k = 1$, $S = p_1$
Pick uniform $[0, 1]$ random variable U
while $U > S$ **do**
  $k = k + 1$, $S = S + p_k$
end while
return k

It is important to note that Algorithm 2 is not the preferred way to sample with replacement. There are faster methods based on binary search and table look up [7, section III] and reservoir algorithms [24]. For our experiments, we used the MATLAB function `randsample`.

2.2. Importance sampling. Monte Carlo algorithms may be best known for their use in approximating continuous functions, such as integrals. Here we make a brief connection from the familiar use of Monte Carlo algorithms in the continuous context to the discrete computation in Algorithm 1. We also describe the purpose of importance sampling, which amounts to the judicious use of nonuniform probabilities.

Define a continuous function f with $f(i) = a_i b_i$, $1 \le i \le n$, and $f(x) \approx 0$ otherwise, and think of the discrete sum $a^T b$ as an integral,

$$a^T b = \sum_{i=1}^{n} a_i b_i = \sum_{i=1}^{n} f(i) \approx \int_0^n f(x)\, dx = f(\zeta)\, n,$$

where the last equality follows from the mean value theorem of integration for some ζ with $0 < \zeta < n$. Approximating $f(\zeta)$ by an average $\frac{1}{c} \sum_{t=1}^{c} f(i_t)$ gives the Monte Carlo estimate

$$\int_0^n f(x)\, dx \approx \frac{n}{c} \sum_{t=1}^{c} f(i_t).$$

For the inner product this implies $a^T b \approx \frac{n}{c} \sum_{t=1}^{c} a_{i_t} b_{i_t}$, which is the approximation produced by Algorithm 1 with uniform probabilities $p_i = 1/n$, $1 \le i \le n$. This is why Algorithm 1 is called a Monte Carlo method.

Running Algorithm 1 with nonuniform probabilities is an instance of *importance sampling*, where the samples are weighted or biased with the goal of reducing the variance [18, section 18], [22, section 1], [23]. To see this, choose a function $p(x)$ that is positive on $[0, n]$ and satisfies $\int_0^n p(x)\,dx = 1$. Then

$$\int_0^n f(x)\,dx = \int_0^n g(x)p(x)\,dx, \qquad \text{where} \qquad g(x) \equiv \frac{f(x)}{p(x)}.$$

Invoking the mean value theorem and approximating by an average gives

$$\int_0^n g(x)p(x)\,dx = g(\zeta) \int_0^n p(x)\,dx = g(\zeta) \approx \frac{1}{c} \sum_{t=1}^{c} g(i_t) = \frac{1}{c} \sum_{t=1}^{c} \frac{f(i_i)}{p(i_t)}.$$

With $p_{i_t} \equiv p(i_t)$, the corresponding approximation for the inner product is

$$a^T b \approx \frac{1}{c} \sum_{t=1}^{c} \frac{a_{i_t} b_{i_t}}{p_{i_t}},$$

which is just the output of Algorithm 3 when run with general probabilities p_i.

For the more general problem of matrix multiplication, Drineas, Kannan, and Mahoney [9, Lemma 4] perform importance sampling by deriving probabilities that minimize the expected value of the normwise absolute error. For the inner product these probabilities reduce to $p_i = |a_i b_i|/|a|^T|b|$. Unfortunately they are not useful here, because they are as expensive to compute as the inner product itself.

2.3. Minimizing the expected value of the variance. Our aim is to choose the probabilities in Algorithm 1 so that some measure of the variance is minimized, in a situation where the vector elements are not known explicitly, and only information of their distribution is known. The application that motivates this approach is described in section 4.

We view the random variable X as a function of the probabilities p and write the variance (2.1) of Algorithm 1 with respect to randomized sampling as

$$\mathbf{Var}[X(p)] = \frac{1}{c} \left(\sum_{i=1}^{n} \frac{a_i^2 b_i^2}{p_i} - (a^T b)^2 \right).$$

We choose probabilities that reduce the variance for all vectors with a particular distribution. That is, we minimize the expected value of the variance,

$$\mathbf{E}_{a,b}[\mathbf{Var}[X(p)]] = \mathbf{E}_{a,b} \left[\frac{1}{c} \left(\sum_{i=1}^{n} \frac{a_i^2 b_i^2}{p_i} - (a^T b)^2 \right) \right], \tag{2.2}$$

where the expected value $\mathbf{E}_{a,b}$ is in regard to the distributions of the elements of a and b.

THEOREM 2.1. *If the vector elements a_i and b_i are independent random variables, $1 \le i \le n$, with finite and nonzero moments $\mathbf{E}[a_i^2 b_i^2]$, then the probability vector q with elements*

$$q_i \equiv \frac{\sqrt{\mathbf{E}_{a,b}[a_i^2 b_i^2]}}{\sum_{j=1}^{n} \sqrt{\mathbf{E}_{a,b}[a_j^2 b_j^2]}}, \qquad 1 \le i \le n, \tag{2.3}$$

minimizes the expected value of the variance in (2.2)*, that is,*

$$E_{a,b}\left[\mathbf{Var}[X(q)]\right] = \min_p \mathbf{E}_{a,b}[\mathbf{Var}[X(p)]],$$

where the minimum ranges over all vectors p *with* $p_j > 0$ *and* $\sum_{j=1}^n p_j = 1$.

Proof. Since the expected value is linear, we can write (2.2) as

$$\mathbf{E}_{a,b}[\mathbf{Var}[X(p)]] = \frac{1}{c}\left(\sum_{i=1}^n \frac{\mathbf{E}_{a,b}\left[a_i^2 b_i^2\right]}{p_i} - \mathbf{E}_{a,b}\left[(a^T b)^2\right]\right).$$

The value $\mathbf{E}_{a,b}\left[(a^T b)^2\right]$ is independent of the probability vector p. Hence the minimal value of $\mathbf{E}_{a,b}[\mathbf{Var}[X(p)]]$ can be obtained by minimizing the function

$$f(p) \equiv \sum_{i=1}^n \frac{\mathbf{E}_{a,b}\left[a_i^2 b_i^2\right]}{p_i}$$

subject to the constraint $g(p) \equiv \sum_{j=1}^n p_j - 1 = 0$. Define the Lagrangian function $L(p,\lambda) \equiv f(p) - \lambda g(p)$, where λ is a scalar, and let q be as in (2.3). From $\mathbf{E}_{a,b}[a_i^2 b_i^2] > 0$ it follows that $q > 0$. Thus $\nabla_p L(q,1) = 0$ and $g(q) = 0$, so that q satisfies the KKT conditions. Since

$$\nabla_p^2 L = \nabla_p^2 f(p) = \operatorname{diag}\left(\frac{\mathbf{E}_{a,b}\left[a_i^2 b_i^2\right]}{p_i^3}\right)$$

is symmetric positive definite for $p > 0$, q is a strictly local minimizer [11, Theorem 9.3.2]. Furthermore, since $\{p|\ p > 0\}$ is an open set, and $\nabla_p^2 f(p)$ is positive definite on this set, the function $f(p)$ is convex, so that q is also a global minimizer [11, section 9.4]. ∎

In the special case when all vector elements have the same distribution, the probabilities in Theorem 2.1 reduce to the uniform probabilities.

COROLLARY 2.2. *If, in addition to the conditions of Theorem* 2.1*, all elements* a_i *and* b_i *are identically distributed, then the expected value of the variance in* (2.2) *is minimized by* $q_i = 1/n$, $1 \le i \le n$.

2.4. Comparison with uniform probabilities. We determine conditions under which the actual variance associated with q is smaller than the variance associated with uniform probabilities $p_u = \frac{1}{n}\begin{pmatrix}1 & \dots & 1\end{pmatrix}^T$. To this end abbreviate

$$V_u \equiv \mathbf{Var}[X(p_u)], \qquad V_q \equiv \mathbf{Var}[X(q)],$$

where the variance V_u is associated with uniform probabilities p_u, and the variance V_q is associated with the optimized probability vector q in (2.3). First we determine the exact difference between the expected values of V_q and V_u.

THEOREM 2.3. *Let* a_i *and* b_i *be independent random variables,* $1 \le i \le n$*, with finite and nonzero moments* $\mathbf{E}[a_i^2 b_i^2]$*, and define*

$$\mu \equiv \frac{1}{n}\sum_{i=1}^n \sqrt{\mathbf{E}[a_i^2 b_i^2]}, \qquad v \equiv \sum_{i=1}^n \left(\sqrt{\mathbf{E}[a_i^2 b_i^2]} - \mu\right)^2.$$

Then

$$\mathbf{E}[V_u] - \mathbf{E}[V_q] = \frac{nv}{c}.$$

Proof. Subtracting the expected values of the variances from (2.2) gives

$$\begin{aligned}\mathbf{E}\left[V_u - V_q\right] &= \frac{1}{c}\left(\sum_{i=1}^{n}\mathbf{E}[a_i^2 b_i^2]\left(n - \frac{\sum_{j=1}^{n}\sqrt{\mathbf{E}[a_j^2 b_j^2]}}{\sqrt{\mathbf{E}[a_i^2 b_i^2]}}\right)\right)\\ &= \frac{n}{c}\left(\sum_{i=1}^{n}\left(\mathbf{E}[a_i^2 b_i^2] - \mu\sqrt{\mathbf{E}[a_i^2 b_i^2]}\right)\right).\end{aligned}$$

From $\sum_{i=1}^{n}\sqrt{\mathbf{E}[a_i^2 b_i^2]} = n\mu^2 = \sum_{i=1}^{n}\mu^2$ it follows that

$$\begin{aligned}\sum_{i=1}^{n}\left(\mathbf{E}[a_i^2 b_i^2] - \mu\sqrt{\mathbf{E}[a_i^2 b_i^2]}\right) &= \sum_{i=1}^{n}\left(\mathbf{E}[a_i^2 b_i^2] - 2\sqrt{\mathbf{E}[a_i^2 b_i^2]}\mu + \mu^2\right)\\ &= \sum_{i=1}^{n}\left(\sqrt{\mathbf{E}[a_i^2 b_i^2]} - \mu\right)^2 = v. \qquad \square\end{aligned}$$

Since $v \geq 0$, Theorem 2.3 confirms again that the expected variance from the optimized probabilities (2.3) is less than that of the uniform probabilities. The value μ can be interpreted as the sample mean of the data $\sqrt{\mathbf{E}\left[a_i^2 b_i^2\right]}$, while $v/(n-1)$ can be viewed as the sample variance. Theorem 2.3 shows that one can expect the variance of the uniform probabilities to be larger than the variance of the optimized probabilities q if the sample variance of the moments $\mathbf{E}_{a,b}[a_i^2 b_i^2]$ is large. If all moments are the same, then $v = 0$ and the optimal probabilities reduce to the uniform probabilities; see Corollary 2.2.

Now we derive an estimate for the relative difference V_q/V_u, because this measure is commonly used in the context of importance sampling [21], [22, section 1.4].

THEOREM 2.4. *Assume that the conditions of Theorem* 2.3 *hold, and for any* δ *with* $0 < \delta < 1$ *define* $\epsilon \equiv \sqrt{\frac{2\mathbf{Var}[V_u]}{\delta\mathbf{E}[V_u]^2}}$. *If*

$$1 - \frac{\mathbf{E}[V_u] - \mathbf{E}[V_q]}{(1+\epsilon)\mathbf{E}[V_u]} + \frac{\sqrt{\frac{2}{\delta}\,\mathbf{Var}[V_u - V_q]}}{(1+\epsilon)\mathbf{E}[V_u]} < f < 1,$$

then with probability at least $1-\delta$ *we have* $V_q \leq f\,V_u$.

Proof. With $\rho \equiv \mathbf{E}[V_u - V_q]$ we can write the probability as

$$\begin{aligned}\mathbf{P}\left[\frac{V_q}{V_u} \leq f\right] &= \mathbf{P}\left[V_q \leq fV_u\right] = \mathbf{P}\left[V_u - V_q \geq (1-f)V_u\right]\\ &= \mathbf{P}\left[V_u - V_q - \rho \geq (1-f)V_u - \rho\right].\end{aligned}$$

We would like to apply Chebyshev's inequality, but at this point it is not possible because V_u on the right-hand side is not constant. To obtain a constant on the right we replace V_u by a little bit more than its expected value and abbreviate the left-hand side by $L \equiv V_u - V_q - \rho$,

$$\mathbf{P}\left[L \geq (1-f)V_u - \rho\right] \geq \mathbf{P}\left[L \geq (1-f)(1+\epsilon)\mathbf{E}[V_u] - \rho\right] - \mathbf{P}\left[V_u \geq (1+\epsilon)\mathbf{E}[V_u]\right].$$

Applying Chebyshev's inequality to the second summand on the right gives

$$\mathbf{P}\left[V_u \geq (1+\epsilon)\mathbf{E}[V_u]\right] \leq \mathbf{P}\left[|V_u - \mathbf{E}[V_u]| \geq \epsilon\,\mathbf{E}[V_u]\right] \leq \frac{\mathbf{Var}[V_u]}{\epsilon^2\,\mathbf{E}[V_u]^2} = \frac{\delta}{2}.$$

Substituting this into the previous inequality gives

$$\mathbf{P}\left[L \geq (1-f)V_u - \rho\right] \geq \mathbf{P}\left[L \geq (1-f)(1+\epsilon)\mathbf{E}[V_u] - \rho\right] - \frac{\delta}{2}.$$

We still have to deal with the first summand on the right. The lower bound for f implies $-\tau \equiv (1-f)(1+\epsilon)\mathbf{E}[V_u] - \rho < 0$, so that

$$\mathbf{P}\left[L \geq -\tau\right] = 1 - \mathbf{P}\left[L \leq -\tau\right] \geq 1 - \mathbf{P}\left[\,|L| \geq \tau\right].$$

Since $L = (V_u - V_q) - \mathbf{E}[V_u - V_q]$ is the difference between a random variable and its expected value, we can now apply Chebyshev's inequality and the lower bound for f to conclude that

$$\mathbf{P}\left[\,|L| \geq \tau\right] \leq \frac{\mathbf{Var}[V_u - V_q]}{\tau^2} \leq \frac{\delta}{2}. \qquad \square$$

The quantity f in Theorem 2.4 represents the relative benefit of the importance sampling method (2.3) over uniform sampling. To be specific, since errors behave as square roots of variances (see (1.1)), $\sqrt{f}$ approximates the fraction of errors from importance sampling as compared to uniform sampling. Moreover, for a given error, the sample size for probabilities (2.3) is reduced by a factor f from that of uniform probabilities. To sum up this comparison of importance sampling (2.3) versus uniform sampling, we see that for a given success probability $1-\delta$, the error is reduced by about a factor of $\sqrt{f}$, while the sample size is reduced by about a factor of f.

Note that

$$1 - \frac{\mathbf{E}[V_u] - \mathbf{E}[V_q]}{\mathbf{E}[V_u]} = \frac{\mathbf{E}[V_q]}{\mathbf{E}[V_u]} \tag{2.4}$$

is the absolute lower bound for f in Theorem 2.4. This agrees with intuition, because we are estimating V_q/V_u. Corollary 2.5 illustrates that there are distributions for which f gets very close to (2.4) for large enough vector dimensions and that in this case $f \approx .36$.

2.5. Numerical experiments on vectors with continuous distributions. We compare the accuracy of Algorithm 1 with the optimized probabilities q_i in (2.3) to that of Algorithm 1 with uniform probabilities for vectors whose elements come from uniform distributions.

First experiment. We use Algorithm 1 to compute an approximation X of $a^T a$, where the elements a_i are independent random uniform $[i, i + i/3]$ variables. The optimized probabilities (2.3) are $q_i = \frac{i^2}{\sum_{j=1}^{n} j^2}$.

Figure 2.1 shows the relative errors $|X - a^T a|/a^T a$ versus sample size c for a single vector a for two versions of Algorithm 1: with uniform probabilities, and with our probabilities q_i from (2.3). The probabilities q_i give an approximation X whose relative error is about a factor 10 smaller than that from uniform probabilities. Sampling about 1% elements with the q_i produces a consistent relative error below 10^{-2}, which means two accurate decimal digits in the output X.

Second experiment. We use Algorithm 1 to compute an approximation X of $a^T b$, where a_i and b_i are independent random uniform $[0, i]$ variables. This is a linearly graded distribution whose interval length increases with the vector dimension n, and it produces values of $f \approx .36$ for sufficiently large n. We first show this analytically.

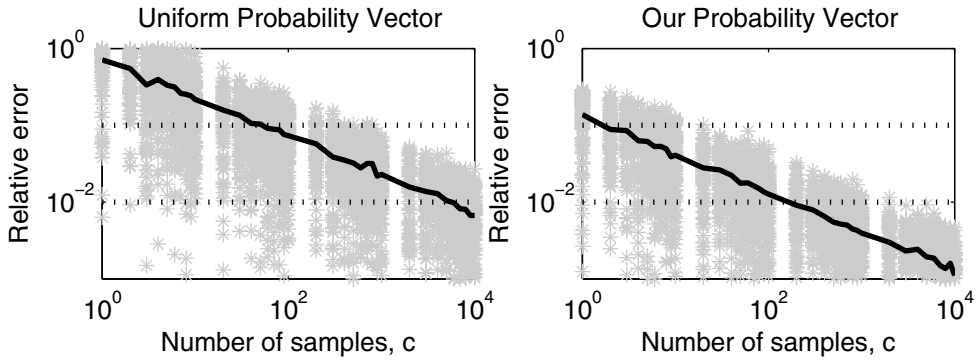

FIG. 2.1. *Relative errors $|X - a^T a|/a^T a$ when the vector elements a_i are independent random uniform $[i, i + i/3]$ variables and $n = 10^6$. For each value of c, the relative errors from 100 trials are shown in light grey and their average in black.*

COROLLARY 2.5. *Let a and b be $n \times 1$ vectors whose elements a_i and b_i are independently distributed uniform $[0, i]$ random variables, $1 \le i \le n$. The optimized probabilities in (2.3) are*

$$q_i = \frac{i^2}{\sum_{j=1}^n j^2} = \frac{6i^2}{n(n+1)(2n+1)}, \qquad 1 \le i \le n.$$

Furthermore, for every $f \ge .36$, $\delta > 0$, and sufficiently large n, we have with probability at least $1 - \delta$ that $V_q \le f\, V_u$.

Proof. We can use the Efron–Stein inequality [5, Proposition 1] to estimate $\mathbf{Var}[V_u] \le \sum_{i=1}^n c_i^2$, where

$$c_i = \max_{a_i, b_i, a_i', b_i'} |V_u(a, b) - V_u(a', b')| \le \frac{n-1}{c}\, i^4 + \frac{n(2n+1)(n+1)}{3c}\, i^2.$$

Properties of the Riemann integral imply, for the first summand,

$$\frac{\sum_{i=1}^n i^4}{n^5} = \frac{\int_1^n x^4 dx + \sum_{i=1}^n i^4 - \int_1^n x^4 dx}{n^5} \le \frac{1}{5} + \frac{1}{n}.$$

Hence $\mathbf{Var}[V_u] = \mathcal{O}(n^{11})$. Similar estimates imply

$$\begin{aligned}
\frac{\mathbf{E}[V_u]}{n^6} &= \frac{1}{cn^6}\left[n \sum_{i=1}^n \mathbf{E}[a_i^2 b_i^2] - \mathbf{E}[(a^T b)^2] \right] \\
&= \frac{1}{cn^6}\left[n \sum_{i=1}^n \mathbf{E}[a_i^2 b_i^2] - \mathbf{E}\left[\sum_{i=1}^n a_i^2 b_i^2 + 2 \sum_{1 \le i < j \le n} a_i b_i a_j b_j \right] \right] \\
&= \frac{1}{cn^6}\left[(n-1) \sum_{i=1}^n \frac{i^4}{9} - 2 \sum_{1 \le i < j \le n} \frac{i^2 j^2}{16} \right] \to \frac{1}{c}\left(\frac{1}{5 \cdot 9} - \frac{1}{16 \cdot 9} \right) \text{ as } n \to \infty.
\end{aligned}$$

This implies $\mathbf{E}[V_u] = \mathcal{O}(n^6)$. Similarly $\mathbf{Var}[V_u - V_q] = \mathcal{O}(n^{11})$. Thus for a given δ we can make ϵ and $\frac{\sqrt{\frac{2}{\delta}\mathbf{Var}[V_u - V_q]}}{(1+\epsilon)\mathbf{E}[V_u]}$ as small as possible by making n large enough and

thus force f to be as close as possible to (2.4). Theorem 2.4 and similar asymptotic estimates as above imply

$$\frac{\mathbf{E}[V_u] - \mathbf{E}[V_q]}{n^6} = \frac{nv}{cn^6} \to \frac{1}{c}\left(\frac{1}{5\cdot 9} - \frac{1}{9^2}\right)$$

and

$$\frac{E[V_u]}{n^6} \to \frac{1}{c}\left(\frac{1}{5\cdot 9} - \frac{1}{16\cdot 9}\right).$$

Hence the limit of (2.4) as $n \to \infty$ is $\frac{\frac{1}{9}-\frac{1}{16}}{\frac{1}{5}-\frac{1}{16}} = .3535\ldots \leq .36$. ∎

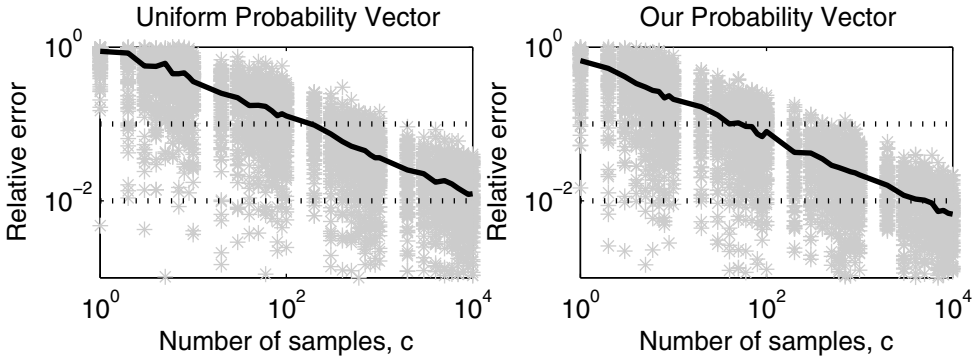

FIG. 2.2. *Relative errors* $|X - a^T b|/|a^T b|$ *when the vector elements* a_i *and* b_i *are independent random uniform* $[0, i]$ *variables and* $n = 10^6$. *For each value of* c, *the relative errors from* 100 *trials are shown in light grey and their average in black.*

The value of $f \approx .36$ from Corollary 2.5 is corroborated by the numerical results displayed in Figure 2.2. There we plot the relative errors $|X - a^T b|/|a^T b|$ versus sample size c for a single pair of vectors a and b for two versions of Algorithm 1: with uniform probabilities, and with the probabilities q_i from Corollary 2.5. The improvement from the probabilities q_i is less dramatic than in the first experiment, which could be due to the high variance of the a_i and b_i. On average, the relative errors associated with the probabilities q_i are approximately a factor of 1.7 smaller than the relative errors associated with uniform probabilities. From Chebyshev's inequality (1.1) it follows that the ratio of variances is approximately the square of the ratio of relative errors, which is about 0.34 and agrees quite well with the theoretical bound of $f \geq 0.36$ from Corollary 2.5.

3. Matrix multiplication. We present an approach for importance sampling for a randomized matrix multiplication algorithm. The approach is motivated by the information retrieval application in section 4 and constructs the probabilities from a discrete distribution. Section 3.1 describes the randomized matrix multiplication algorithm, and section 3.2 discusses the importance sampling for this algorithm.

3.1. The algorithm. The multiplication of an $m \times n$ matrix A with an $n \times k$ matrix B can be viewed as a sequence of inner products involving the rows of A and the columns of B. Running Algorithm 1 on each inner product gives an $m \times k$ matrix C whose elements $C_{ij} \approx A_{(i)}B^{(j)}$ are created independently by sampling elements from $A_{(i)}$ and $B^{(j)}$. The resulting algorithm is the matrix multiplication

algorithm from [8, section 5] (see Algorithm 3); it is a generalization of the BASICMATRIXMULTIPLICATION algorithm in [9, Figure 2]. In contrast to Algorithm 3, the BASICMATRIXMULTIPLICATION algorithm in [9, Figure 2] selects whole columns of A and whole rows of B. This amounts to first sampling the indices $\{i_1, \ldots, i_c\}$ and then using them for all mk inner products.

ALGORITHM 3. Matrix Multiplication Algorithm [8]

Input: real $m \times n$ matrix A, real $n \times k$ matrix B
probabilities $p = \begin{pmatrix} p_1 & \ldots & p_n \end{pmatrix}^T$ with $p_i > 0$ and $\sum_{i=1}^{n} p_i = 1$
integer c, where $1 \le c \le n$
Output: $m \times k$ matrix C that approximates AB

$C_{ij} = 0,\ 1 \le i \le m,\ 1 \le j \le k$
for $i = 1 : m,\ j = 1 : k$ **do**
  for $t = 1 : c$ **do**
    Sample i_t from $\{1, \ldots, n\}$ with probability p_{i_t}
      independently and with replacement
    $C_{ij} = C_{ij} + \frac{A_{i,i_t} B_{i_t,j}}{c\, p_{i_t}}$
  end for
end for
return C

3.2. Importance sampling. Algorithm 3 samples each inner product according to the same probability vector, which consists of n probabilities. Our approach for defining these probabilities is motivated by the information retrieval application in section 4. There the $m \times n$ matrix A is a term-by-document matrix whose m rows represent documents, and each column represents a term. The k columns of the matrix B represent query vectors. Following the notation of [9, section 2], we distinguish the rows $A_{(i)}$ of A and the columns $B^{(j)}$ of B,

$$A = \begin{pmatrix} - & A_{(1)} & - \\ - & A_{(2)} & - \\ & \vdots & \\ - & A_{(m)} & - \end{pmatrix}, \qquad B = \begin{pmatrix} | & | & & | \\ B^{(1)} & B^{(2)} & \ldots & B^{(k)} \\ | & | & & | \end{pmatrix}.$$

We define discrete probability distributions by assuming that each of the m rows of A (documents) is equally likely to be chosen and so is, independently, each of the k columns of B (query vectors). The expected values for the squared elements of A and B with regard to this distribution are therefore

$$\mathbf{E}_A\left[A_{ij}^2\right] = \frac{1}{m} \sum_{t=1}^{m} A_{tj}^2 = \frac{\|A^{(j)}\|_2^2}{m}, \qquad 1 \le i \le m, \quad 1 \le j \le n, \tag{3.1}$$

$$\mathbf{E}_B\left[B_{jl}^2\right] = \frac{1}{k} \sum_{t=1}^{k} B_{jt}^2 = \frac{\|B_{(j)}\|_2^2}{k}, \qquad 1 \le j \le n, \quad 1 \le l \le k. \tag{3.2}$$

Putting these expressions into the optimized probabilities (2.3) gives

$$q_j = \frac{||A^{(j)}||_2 ||B_{(j)}||_2}{\sum_{t=1}^{n} ||A^{(t)}||_2 ||B_{(t)}||_2}, \qquad 1 \le j \le n. \tag{3.3}$$

The probabilities q can be computed in $\mathcal{O}(n(k+m))$ arithmetic operations. Algorithm 3 needs an additional $\mathcal{O}(ckm)$ operations to approximate the matrix product. This can be cheaper than a deterministic matrix multiplication algorithm when n is large. Note that the probabilities (3.3) happen to be identical to the probabilities that minimize the expected value of the absolute normwise error of the BasicMatrixMultiplication [9, Lemma 4].

4. Application to information retrieval. We show how to apply Algorithm 3 to an information retrieval problem with Reuters documents and Wikipedia web pages and how to efficiently compute the probabilities (3.3) for a discrete data set. Section 4.1 describes our assumptions. Numerical experiments are presented for the Reuters documents in section 4.2 and for the Wikipedia web pages in section 4.3.

4.1. Assumptions. The application is a nearest neighbor problem of the following type [3, section 1]:

> *Given a collection of data points and a query point in an m-dimensional metric space, find the data point that is closest to the query point.*

The data points are documents, and the goal is find those documents that best match a query submitted by a user. More generally, we want to determine the k documents that best match the query.

Cosines. In the so-called *vector-space* model [2, section 2.1], the document collection is represented by a *term-by-document* matrix[2] A, where each column of A is associated with a term, each row $A_{(i)}$ is associated with a document, and an element A_{ij} corresponds to the relative frequency of term j in document i. The amount of "matching" between a document vector $A_{(i)}$ and a query vector b is determined by the cosine of the angle between the two vectors, $\frac{A_{(i)}b}{\|A_{(i)}\|_2\|b\|_2}$.

Ordinal rank. What matters in this application is not the actual values of the cosines but only the induced ordinal rank [25, Definition 2.1]. We apply three criteria to evaluate the accuracy of the ranking. In a top-k list ranking, we want to determine the k documents that best match b and rank them according to their nearness to b, while in a top-k bucket ranking, we also want the k best documents but do not care about their ordering; see [25, section 3]. Finally, in a top-k-out-of-l bucket ranking, one wants to know whether the top k documents are contained in a top bucket of size l.

Online/offline computations. We assume that certain computations can be performed "offline," while others need to take place "online." *Online algorithms* receive a sequence of requests and immediately perform an action in response to each request, while *offline algorithms* receive the entire sequence of requests in advance [15, section 1]. In this application, the information associated with the document matrix A, such as distributions of terms and probabilities, has been computed offline, while the process of matching documents with a query vector is performed online.

Query matching. The process of determining documents $A_{(i)}$ that match a query vector b consists of a matrix vector multiplication Ab performed by Algorithm 3.

The idea is to exploit the offline availability of the term-by-document matrix A and to compute a probability vector that covers all m documents. Since column j of A represents the frequencies of term j in the m different documents, we can view element j in any document $A_{(i)}$ of A as coming from the discrete distribution defined

[2]For simplicity, we define the term-by-document matrix as the transpose of the one in [2, section 2.1].

by column j, that is,

$$\mathbf{E}_A[A_{ij}^2] = \sum_{t=1}^{m} p_t\, A_{tj}^2, \qquad 1 \le j \le n.$$

If a term is equally likely to occur in any of the m documents, we can set $p_t = 1/m$ so that $\mathbf{E}_A[A_{ij}^2] = \|A^{(j)}\|_2^2/m$, which is just (3.1). Treating the query b as a constant gives $\sqrt{\mathbf{E}_b[b_j^2]} = |b_j|$ in (3.2). Independence of the documents and the query implies

$$\mathbf{E}_{A,b}[A_{ij}^2 b_j^2] = \mathbf{E}_A[A_{ij}^2]\,|b_j|^2 = \|A^{(j)}\|_2^2\,|b_j|^2, \qquad 1 \le j \le n.$$

Hence the probabilities for the inner product $A_{(i)}b$ are

$$q_j = \frac{\|A^{(j)}\|_2\,|b_j|}{\sum_{t=1}^{n} \|A^{(t)}\|_2\,|b_t|}, \qquad 1 \le j \le n, \tag{4.1}$$

which is (3.3) for the special case of matrix vector multiplication.

The probabilities q_j in (4.1) are used for all m documents, which means that one needs to compute only a single probability vector for each query vector b. Since the column norms $||A^{(t)}||_2$ have been computed offline, computing the probabilities for a particular query vector b requires only $\mathcal{O}(n)$ online operations, and the online computation of q does not depend on the number of documents. As a consequence, Algorithm 3 approximates the matrix vector product Ab with only $\mathcal{O}(n + cm)$ online operations, where $c \ll n$ is the number of elements sampled. Compared to a deterministic computation with $\mathcal{O}(mn)$ arithmetic operations, this reduction in operation count is especially important when the collection contains many more terms than documents, i.e., $n \gg m$.

4.2. Numerical experiments with the Reuters data set. The term-by-document matrix A for the Reuters Transcribed Subset data set[3] is a subset of the well-known Reuters-21578 collection.[4] The subset contains 201 documents and 5601 terms, so the matrix A is 201×5601. We use two different query vectors: (1) a normalized version of the sum of three rows of A, and (2) a single row of A.

For the first query vector we determine a top-25 bucket ranking to illustrate that probabilities (4.1) can be substantially more accurate than uniform probabilities. Figure 4.1 compares the ranking computed by the deterministic inner product to that computed by two versions of Algorithm 3: one that samples with uniform probabilities, and a second version that samples with probabilities (4.1). Algorithm 3 uses $c = 56$, where c/n amounts to 1% of the elements. Since the vectors are sparse, this is not a perfect measure of the reduction in work, but we believe that it is still meaningful. We chose the uniform probabilities $p_i = 1/n_z$ for $b_i \neq 0$, where n_z is the number of nonzero elements in b, because the probabilities $p_i = 1/n$ gave very poor results. Figure 4.1 shows that Algorithm 3 with uniform probabilities finds only 11 documents in the top-25 bucket and one correct ranking, while Algorithm 3 with probabilities (4.1) finds 22 documents and six correct rankings.

Tables 4.1 and 4.2 present detailed comparisons between uniform probabilities and probabilities (4.1). We ran 100 trials with each of the two query vectors and

[3] http://archive.ics.uci.edu/ml/datasets/Reuters+Transcribed+Subset.

[4] http://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection.

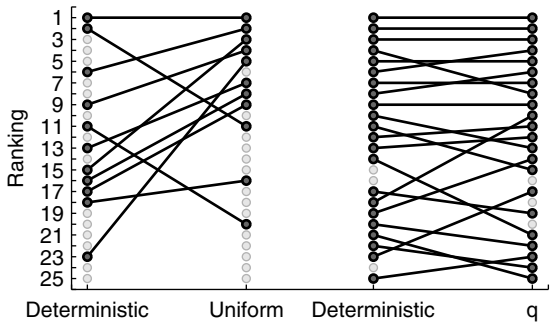

FIG. 4.1. *Top-25 bucket ranking. Left columns of the two graphs: Ranking computed by the deterministic inner product. Right columns: Rankings computed by Algorithm 3 with $c = 56$. Left graph: Algorithm 3 with uniform probabilities. Right graph: Algorithm 3 with probability vector q from (4.1). Corresponding documents are connected by lines. A horizontal line means that Algorithm 3 has ranked the associated document correctly. Light grey dots represent documents that are not present in the opposite list.*

TABLE 4.1
Table entries give the number of correct rankings in 100 runs with the first query vector when Algorithm 3 samples $c = 56$ elements.

k	Top-k list		Top-k bucket		Top-k-out-of-25 bucket	
	(4.1)	Uniform	(4.1)	Uniform	(4.1)	Uniform
1	69	9	69	9	100	54
2	50	1	65	1	100	20
3	30	0	56	0	100	0
5	3	0	15	0	100	0
10	0	0	6	0	99	0

TABLE 4.2
Table entries give the number of correct rankings in 100 runs with the second query vector when Algorithm 3 samples $c = 56$ elements.

k	Top-k list		Top-k bucket		Top-k-out-of-25 bucket	
	(4.1)	Uniform	(4.1)	Uniform	(4.1)	Uniform
1	81	10	81	10	100	56
2	63	1	77	2	100	28
3	52	0	80	0	100	8
5	4	0	25	0	100	2
10	0	0	20	0	99	0

listed the number of correct rankings of each type. In both cases, Algorithm 3 sampled $c = 56$ elements, where c/n corresponds to 1% of the elements. Table 4.1 illustrates that both types of probabilities tend to produce less accurate rankings the more documents they are asked to rank. However, probabilities (4.1) can result in substantially more accurate rankings than uniform probabilities. In particular, the 25 documents determined by probabilities (4.1) contain the top-10 documents 99% of the time. The uniform probabilities perform much worse for the first query vector (see Table 4.1) than for the second (see Table 4.2). This may be due to the sparsity of the second query vector. Furthermore, the dimensions of this Reuters data set are rather small.

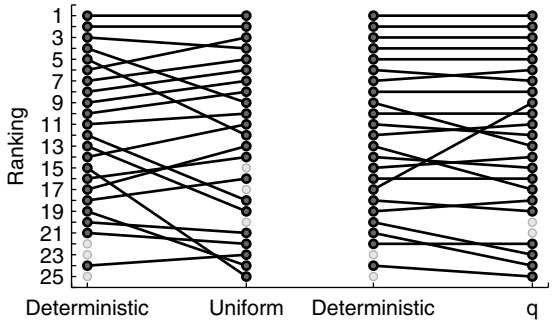

FIG. 4.2. *Top-25 list ranking. Left columns of the two graphs: Ranking computed by the deterministic inner product. Right columns: Rankings computed by Algorithm 3 with $c = 2000$. Left graph: Algorithm 3 with uniform probabilities. Right graph: Algorithm 3 with probabilities q from (4.1). Corresponding feature vectors are connected by lines. A horizontal line means that Algorithm 3 has ranked the associated feature vector correctly. Light grey dots represent feature vectors that are not present in the opposite list.*

4.3. Experiments with Wikipedia data set. We extracted a term-by-document matrix B from Wikipedia web pages[5] and performed a nonnegative matrix factorization [16] to obtain an approximate factorization $B \approx HA$, where A is an $m \times n$ nonnegative matrix.

The m rows of A can be interpreted as "feature vectors" [20] that represent the data set B. The rows of B are the query vectors. The matrix B has $n = 198853$ columns and 203688 rows, while A has $n = 198853$ columns and $m = 200$ rows. This gives $m = 200$ feature vectors, each of dimension $n = 198853$.

As in the previous experiments, we determine a top-25 list ranking for a single query vector. Figure 4.2 compares the ranking computed by the deterministic inner product to that computed by two versions of Algorithm 3. Algorithm 3 uses $c = 2000$, where c/n amounts to 1% of the elements. Figure 4.2 shows that uniform probabilities produce only two correct rankings, while probabilities (4.1) produce nine correct rankings.

Tables 4.3 and 4.4 present more detailed comparisons between uniform probabilities and probabilities (4.1). We ran 100 trials with the same query vector and list the number of correct rankings of each type. In Table 4.3, Algorithm 3 samples $c = 2000$ elements, where c/n amounts to 1% of the elements. As before, probabilities (4.1) can result in substantially more accurate rankings than uniform probabilities. In particular, the 25 documents selected by probabilities (4.1) contain the top-10 documents.

Table 4.4 illustrates what happens when Algorithm 3 samples only half as many elements, i.e., $c = 1000$, where c/n corresponds to only 0.5% of the elements. Both probabilities produce fewer accurate rankings than for $c = 2000$. However, probabilities (4.1) still manage to find the top-10 documents in a bucket of 25.

Figure 4.3 illustrates how the accuracy of rankings produced by Algorithm 3 with probabilities (4.1) depends on the amount of sampling c. For $c = 2000$, a top-10 bucket contains on average the top-9 documents. Sampling more does not seem to help a lot. However, sampling less definitely reduces the number of correctly ranked documents.

[5] http://en.wikipedia.org/wiki/Wikipedia:Database_download.

TABLE 4.3

Table entries give the number of correct rankings in 100 *runs with the same query vector when Algorithm* 3 *samples* $c = 2000$ *elements.*

k	Top-k list		Top-k bucket		Top-k-out-of-25 bucket	
	(4.1)	Uniform	(4.1)	Uniform	(4.1)	Uniform
1	97	64	97	64	100	99
2	97	33	100	53	100	99
3	74	15	75	29	100	99
5	29	1	49	9	100	92
10	0	0	28	1	100	79

TABLE 4.4

Table entries give the number of correct rankings in 100 *runs with the same query vector when Algorithm* 3 *samples* $c = 1000$ *elements.*

k	Top-k list		Top-k bucket		Top-k-out-of-top-25 bucket	
	(4.1)	Uniform	(4.1)	Uniform	(4.1)	Uniform
1	91	54	91	54	100	98
2	85	23	93	37	100	91
3	54	3	60	9	100	91
5	12	0	33	4	100	79
10	0	0	13	0	100	43

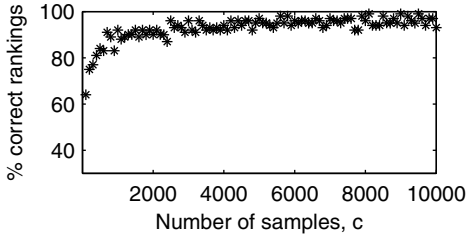

FIG. 4.3. *Average percentage of correct rankings in a top-*10 *bucket ranking over* 10 *trials produced by Algorithm* 3 *with probabilities* (4.1) *as a function of sample size* c*, where* $c = 100, 200, \ldots, 10000$*.*

Overall, the greater accuracy with probabilities (4.1) compared to uniform probabilities is more striking in the Reuters data set. This could be due to the difference in the dimension or sparsity of the vector size or to the distribution of elements.

5. Conclusions. We performed importance sampling for the randomized matrix multiplication algorithm from [8, 9] by exploiting the distributions of the matrix elements. As a result, we were able to derive probabilities that minimize the expected value of the variance for inner products. Information retrieval problems on Reuters and Wikipedia data sets illustrate how to compute these probabilities for discrete distributions and show that the computation is efficient in an online/offline setting.

Our experiments indicate that the Monte Carlo algorithms for matrix multiplication have low relative accuracy: They tend to produce only 1–2 accurate decimal digits, regardless of the amount of sampling. More specifically, the minimal amount of sampling to achieve this accuracy is about 1% of the elements, but any more sampling does not seem to improve the accuracy. In spite of their low relative accuracy, though,

the algorithms can be useful in applications where exact values are not required, such as ordinal ranking in information retrieval applications.

We believe that the idea of minimizing expected variance merits further study for other randomized matrix algorithms. Many of these algorithms are Monte Carlo algorithms that rely on sampling and thus could benefit from efficient sampling probabilities.

Acknowledgments. We thank Dianne O'Leary and Petros Drineas for helpful discussions.

REFERENCES

[1] M. Belabbas and P. J. Wolfe, *On sparse representations of linear operators and the approximation of matrix products*, ArXiv e-prints, 2007.

[2] M. W. Berry, Z. Drmač, and E. R. Jessup, *Matrices, vector spaces, and information retrieval*, SIAM Rev., 41 (1999), pp. 335–362.

[3] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, *When is "nearest neighbor" meaningful?*, in Proceedings of the 7th International Conference on Database Theory, Springer, Berlin, 1999, pp. 217–235.

[4] E. Bingham and H. Mannila, *Random projection in dimensionality reduction: Applications to image and text data*, in KDD '01: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, 2001, pp. 245–250.

[5] S. Boucheron, G. Lugosi, and P. Massart, *Concentration inequalities using the entropy method*, Ann. Probab., 31 (2003), pp. 1583–1614.

[6] E. Cohen and D. D. Lewis, *Approximating matrix multiplication for pattern recognition tasks*, in Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 1997, pp. 682–691.

[7] L. Devroye, *Non-uniform Random Variate Generation*, Springer-Verlag, New York, 1986.

[8] P. Drineas and R. Kannan, *Fast Monte-Carlo algorithms for approximate matrix multiplication*, in Proceedings of the 42nd IEEE Symposium on Foundations of Computer Sciences, IEEE Computer Society, Los Alamitos, CA, 2001, pp. 452–459.

[9] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices* I: *Approximating matrix multiplication*, SIAM J. Comput., 36 (2006), pp. 132–157.

[10] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices* II: *Computing a low-rank approximation to a matrix*, SIAM J. Comput., 36 (2006), pp. 158–183.

[11] R. Fletcher, *Practical Methods of Optimization*, 2nd ed., John Wiley & Sons, Chichester, 1987.

[12] A. Frieze, R. Kannan, and S. Vempala, *Fast Monte-Carlo algorithms for finding low-rank approximations*, J. ACM, 51 (2004), pp. 1025–1041.

[13] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss, *Near-optimal sparse Fourier representations via sampling*, in STOC '02: Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing, ACM, New York, 2002, pp. 152–161.

[14] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.

[15] R. M. Karp, *On-line algorithms versus off-line algorithms: How much is it worth to know the future?*, in Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture, Amsterdam, The Netherlands, 1992, North–Holland, Amsterdam, 1992, pp. 416–429.

[16] D. D. Lee and H. S. Seung, *Algorithms for Non-negative Matrix Factorization*, Adv. Neural Info. Proc. Syst. 13, MIT Press, Cambridge, MA, 2000, pp. 556–562.

[17] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press, New York, 2006.

[18] D. P. O'Leary, *Scientific Computing with Case Studies*, SIAM, Philadelphia, 2009.

[19] T. Sarlós, *Improved approximation for large matrices via random projections*, in Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Sciences, IEEE Computer Society, Los Alamitos, CA, 2006, pp. 143–152.

[20] F. Shahnaz, M. W. Berry, V. P. Pauca, and R. J. Plemmons, *Document clustering using nonnegative matrix factorization*, Inform. Process. Manag., 42 (2006), pp. 373–386.

[21] P. J. Smith, M. Shafi, and H. Gao, *Quick simulation: A review of importance sampling techniques in communications systems*, IEEE J. Sel. Areas Commun., 15 (1997), pp. 597–613.
[22] R. Srinivasan, *Importance Sampling: Applications in Communication and Detection*, Springer-Verlag, Berlin, 2002.
[23] S. T. Tokdar and R. E. Kass, *Importance sampling: A review*, WIREs Comp. Stat., 2 (2010), pp. 54–60.
[24] J. S. Vitter, *Random sampling with a reservoir*, ACM Trans. Math. Software, 11 (1985), pp. 37–57.
[25] R. S. Wills and I. C. F. Ipsen, *Ordinal ranking for Google's PageRank*, SIAM J. Matrix Anal. Appl., 30 (2009), pp. 1677–1696.
[26] J. Zou, *A sublinear algorithm for the recovery of signals with sparse Fourier transform when many samples are missing*, Appl. Comput. Harmon. Anal., 22 (2007), pp. 61–77.
[27] J. Zou, A. Gilbert, M. Strauss, and I. Daubechies, *Theoretical and experimental analysis of a randomized algorithm for sparse Fourier transform analysis*, J. Comput. Phys., 211 (2006), pp. 572–595.