Monte Carlo Methods and Their Applications in Big Data Analysis

Chapter · January 2016

DOI: 10.1007/978-3-319-25127-1_7

2 authors:

Hao Ji
California State Polytechnic University, Pomona
14 PUBLICATIONS **54** CITATIONS
SEE PROFILE

Yaohang Li
Old Dominion University
117 PUBLICATIONS **671** CITATIONS
SEE PROFILE

Monte Carlo Methods and their Applications in Big Data Analysis

Hao Ji and Yaohang Li

1 Introduction

Numerical methods known as Monte Carlo methods can be loosely defined in general terms to be any methods that rely on random sampling to estimate the solutions. Monte Carlo methods are often applied to problems which are either too complicated to be described by a mathematical model or whose parameter space is too large to be explored systematically.

Recent years have witnessed dramatic increase of data in many fields of science and engineering [10], due to the advancement of sensors, mobile devices, biotechnology, digital communication, and internet applications. These massive, continuously growing, complex, and diverse data sets are often referred to as the big data. Analyzing big data demands cost-effective and innovative forms of information processing for enhanced insight and decision making. While many traditional, deterministic data analysis methods have great difficulties to scale to the massive big data sets, Monte Carlo methods, which are based on random sampling techniques, become important and powerful tools for big data applications. Particularly, for certain big data sets beyond the computational capability of the most powerful supercomputers, Monte Carlo data analysis methods using random sampling are the only viable approaches.

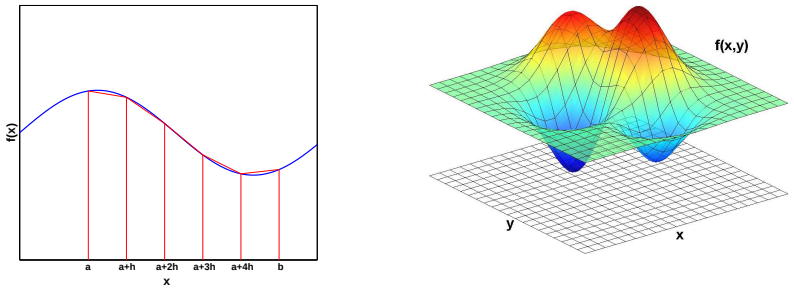
This article starts with a basic description of the principles of Monte Carlo methods. It then discusses the variance reduction techniques, including stratified sampling, control variates, antithetic variates, and importance sampling, to design

Hao Ji and Yaohang Li
Department of Computer Science
Old Dominion University
Norfolk, VA 23529, USA
e-mail: {hji,yaohang}@cs.odu.edu
H. Ji, Y. Li, Monte Carlo Methods and their Applications in Big Data Analysis, Mathematical Problems in Data Science - Theoretical and Practical Methods, Springer, ISBN: 978-3-319-25127-1, 2015.

"smart" Monte Carlo. Examples of Monte Carlo methods on estimating sum of a large array, solving linear systems, multiplying matrix products, recovering missing matrix, and approximating low-rank matrices. Although in order to keep the presentation simple, these application examples are presented in a relatively small scale, the underlying Monte Carlo techniques can be effectively extend to big data sets.

2 The Basic of Monte Carlo

Monte Carlo methods provide approximate solutions to a variety of mathematical problems by random sampling. Let us take the numerical integration as an example, which constitutes a broad family of algorithms in numerical analysis. Suppose we want to calculate a one-dimensional definite numerical integral, $I = \int_a^b f(x)\,dx$. A common numerical integral method is to divide the one-dimensional interval into N subintervals and then to sum the area corresponding to each subinterval using either rectangular, trapezoidal, or Simpson's rules (Fig. 1(a)) [1]. Similarly, for two-dimensional intervals, the number of 2D subintervals becomes N^2 (Fig. 1(b)). In general, for d-dimensional integration problems, the d-dimensional space needs to be divided into N^d subintervals. For a not very high dimensional problem with $d = 20$ and $N = 100$, the total number of subintervals that need to be evaluated goes up to 10^{40}, which is unapproachable by most numerical integration algorithms. Mathematically, this is referred to as the "curse of dimensionality."



(a) 1D integral (b) 2D integral

Fig. 1 Numerical Integration using Deterministic Methods

In contrast, Monte Carlo methods estimate the integral by statistical sampling techniques [9]. Let us consider a one-dimensional integral $I_{0-1} = \int_0^1 f(x)\,dx$, which can be easily extended to a more general integral of $I = \int_a^b f(x)\,dx$. Suppose that the random variables $x_1, x_2, \cdots, x_N$ are drawn independently from the probability

density function $p(x)$. A function F may be defined as

$$F = \sum_{i=1}^{N} f(x_i)\, p(x_i).$$

The expectation value of F becomes

$$E(F) = \int_0^1 f(x)\, p(x)\, dx.$$

The crude Monte Carlo integration method assumes that the probability density function $p(x)$ is uniform, i.e., the random samples $f(x_1), f(x_2), \cdots, f(x_N)$ are equally important, and then

$$E(F) = \int_a^b f(x)\, dx.$$

Correspondingly, the variance of F becomes

$$Var(F) = \frac{1}{N} \int_0^1 (f(x) - E(F))^2\, dx = \frac{1}{N}\sigma^2,$$

where σ^2 is the inherent variance of the integrant function $f(x)$. Clearly, we can find that the standard deviation of the estimator θ is $\sigma N^{-1/2}$. This means that as $N \to \infty$, the distribution of F narrows around its mean at the rate of $O\left(N^{-1/2}\right)$.

Now, let us extend the Monte Carlo integration method to a d-dimensional integral $I_d = \int_0^1 \cdots \int_0^1 f(x)\, dx$, the expectation of $F_d = \sum_{i=1}^{N} f(x_i)/N$ on uniformly distributed random variable vectors $x_1, x_2, \cdots, x_N$ becomes

$$E(F_d) = \int_0^1 \cdots \int_0^1 f(x)\, dx = I_d.$$

The variance of the estimator F_d is σ_d^2/N, where σ_d^2 is the inherent variance of the integrant function $f(x)$. If an integral function $f(x)$ is given, σ_d^2 is a constant and therefore, similar to one-dimensional integral, the convergence rate of Monte Carlo is $O\left(N^{-1/2}\right)$, which is independent of dimensionality.

In summary, compared to the deterministic numerical integration methods, whose convergence rate is $O(N^{-\alpha/d})$, where α is the algorithm related constant and d is the dimension, Monte Carlo integration method yields a convergence rate of $O(N^{-1/2})$ [16], which can somehow avoid the "curse of dimensionality." Moreover, computations on each random samples are independent, which can be carried out in an embarrassingly parallel manor to harness the power of large-scale parallel and distributed computing architectures [14, 15]. On the other hand, the main disadvantage of Monte Carlo is that the convergence of Monte Carlo methods is very slow, roughly for every one digit of accuracy usually requiring 100 times more computations.

3 Variance Reduction

Crude Monte Carlo treat all random samples are equally important. In reality, we can often gain additional knowledge from the application domain, which can be taken advantage to come up with better estimators. Variance reduction is a procedure of deriving an alternative estimator to obtain a smaller variance than the crude Monte Carlo estimator and improve the precision of the Monte Carlo estimates for a given number of samples. In practical applications, a good estimator leading to million times more accurate than a bad one is not rarely seen. In this section, we describe some of the popular variance reduction techniques [9, 16], including stratified sampling, control variates, antithetic variates, and importance sampling. These variance reduction methods, if appropriately used, can significantly improve the efficiency of Monte Carlo methods in processing and analyzing big data sets.

3.1 Stratified Sampling

The inherent variance of the integral function $f(x)$ may vary significantly in different regions. The fundamental idea of stratified sampling is to apply more samples in the region with large variability and vice versa. Using the one-dimensional integral in interval $[0,1]$ as an example, stratified sampling is to partition the interval $[0,1]$ into several disjoint subinterval ranges i.e., $[0,\alpha_1),[\alpha_1,\alpha_2),\cdots,[\alpha_{k-1},\alpha_k),[\alpha_k,1]$. Then a Monte Carlo estimator $F_{\alpha_{i-1},\alpha_i}$ is applied to each range $[\alpha_{i-1},\alpha_i)$ separately so that

$$F_{\alpha_{i-1},\alpha_i} = \sum_{j=1}^{n_i} (\alpha_i - \alpha_{i-1}) \frac{1}{n_i} f(x_j),$$

where x_j is a random sample in $[\alpha_{i-1},\alpha_i)$ and n_i is the total number of samples. Clearly,

$$E\left(F_{\alpha_{i-1},\alpha_i}\right) = \int_{\alpha_{i-1}}^{\alpha_i} f(x)dx.$$

The overall stratified sampling estimator $F_{stratified}$ is the combination of all estimators in different subinterval ranges

$$F_{stratified} = \sum_{i=1}^{k+1} F_{\alpha_{i-1},\alpha_i},$$

which is an unbiased estimator of the integral. The variance of the stratified sampling estimator becomes

$$Var\left(F_{stratified}\right) = \sum_{i=1}^{k+1} \left(\alpha_i - \alpha_{i-1)}\right) \frac{1}{n_i} \sigma^2_{\alpha_{i-1},\alpha_i},$$

where $\sigma^2_{\alpha_{i-1},\alpha_i}$ is the variance of the integrant function $f(x)$ in subinterval $[\alpha_{i-1},\alpha_i)$. This reveals the rationale of stratified sampling, which is to minimize $\frac{1}{n_i}\sigma^2_{\alpha_{i-1},\alpha_i}$ in each subinterval. If stratification is well carried out, the variance of stratified sampling will be less than that of crude Monte Carlo.

3.2 Control Variates

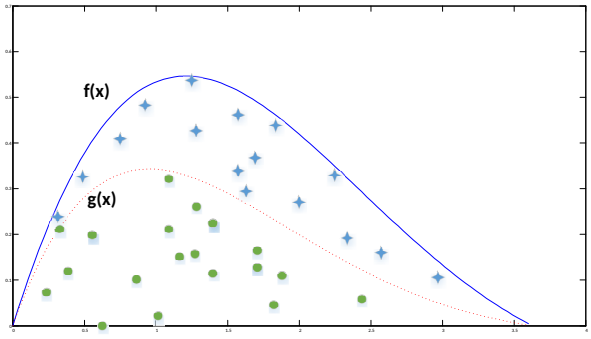


Fig. 2 Control variates approach. If $g(x)$ can be estimated theoretically, the sampling errors are limited to the area of $f(x)-g(x)$

If there is another estimator which is positively correlated to the one to be evaluated and can be easily calculated with known expectation, it can be used as a control variate. For example, in the Monte Carlo integration example, the original integral $I_{0-1}=\int_0^1 f(x)dx$ can be rewritten into a summation of two parts

$$I_{0-1}=\int_0^1 g(x)dx+\int_0^1 (f(x)-g(x))\,dx,$$

where $g(x)$ is a simple function that can be either integrated theoretically or easily calculated and absorbs most of the variation of $f(x)$ by mimicking $f(x)$ as shown in Fig. 2, we only need to estimate the difference between $f(x)-g(x)$, which will result in variance reduction of the estimator.

3.3 Antithetic variates

Contrast to the control variates method based on a strongly positively-correlated estimator with F, the antithetic variates method takes advantage of an estimator F' that is strongly negatively correlated with F. Suppose that F is an unbiased estimator

of a quantity, $E(F) = E(F')$, and $Cov(F,F') < 0$. Then, the antithetic estimator $F_{antitheticc} = (F+F')/2$ is also an unbiased estimator of this quantity. The sampling variance of $F_{antitheticc}$ becomes

$$Var(F_{antitheticc}) = \frac{Var(F)}{v} + \frac{Var(F')}{4} + \frac{Cov(F,F')}{2}.$$

Here $Cov(F,F') < 0$ due to negative correlation between F and F', which leads to overall smaller variance in $F_{antitheticc}$ than that of F and F'.

3.4 Importance Sampling

The importance sampling technology is often used in statistical resampling, which reduces variance by emphasizing the sampling on regions of interest. For example, by introducing a new proposal function $g(x)$, the original integral $I_{0-1} = \int_0^1 f(x)dx$ can be rewritten as

$$I_{0-1} = \int_0^1 \frac{f(x)}{g(x)} g(x)dx = \int_0^1 \frac{f(x)}{g(x)} dG(x),$$

where $G(x)$ is a cumulative density function (CDF). $f(x)/g(x)$ is called the likelihood ratio. With random samples drawn from a proposal distribution whose CDF is $G(x)$ instead of sampling from a uniform distribution, the variance of the importance sampling estimator $F_{importance\ sampling}$ becomes

$$Var\left(F_{importance\ sampling}\right) = \int_0^1 \left(\frac{f(x)}{g(x)} - E(F)\right)^2 dG(x).$$

A good likelihood ratio can result in a significant variance reduction.

In practices, assume that we know nothing about the target distribution at the very beginning, we may have to start from uniform sampling. However, after initial sampling, we have a better understanding of the target distribution, which results in a better proposal function. The resampling can be guided by the new proposal function and leads to a better approximation of the target distribution.

4 Examples of Monte Carlo Methods in Data Science

In this section, we present several application examples as case studies of using Monte Carlo methods for data analysis. Variance reduction techniques are applied to build "smart" Monte Carlo estimators to enhance sampling efficiency.

4.1 Case Study 1: Estimation of Sum

In many practical data analysis applications, we are often required to estimate the sum of a large dataset. However, due to many reasons, we are not allowed to visit every element in the dataset but want to get a good estimation. The following is an application example to estimate the overall salary expense in a company.

Consider a big, global company having $7,140$ employees falling in the following categories, 78 managers, $4,020$ engineers, $2,008$ salesmen, and $1,034$ technicians. Now we want to estimate the overall expense in employee salaries in this company. Due to cost as technical difficulty, we are only allowed to use 100 samples.

The simplest sampling method is the crude Monte Carlo using uniform sampling without considering different categories. The main problem of uniform sampling is ignoring the differences among categories, which will thus lead to a large estimation variance $460,871K \pm 80,712K$. More importantly, the percentage of managers is around 1% in all employees. There is a high chance that the selected 100 samples may miss the manager category. Stratified sampling can better address this problem. We can calculate the number of samples falling into each category, specifically, $78/7140 \times 100 = 1$ sample in managers, $4020/7140 \times 100 = 56$ samples in engineers, $2008/7140 \times 100 = 28$ samples in salesmen, and $1034/7140 \times 100 = 15$ samples in technicians. As a result, stratified sampling yields better estimation $(500,112K \pm 30,147K)$ than that of uniform sampling.

If we have additional information, we can construct even better stratified sampling estimator. If we happen to know about the salary variation in each employee category (for example, from previous years' data), i.e., Managers $(200K \sim 500K)$, Salesmen $(40K \sim 120K)$, Engineers $(60K \sim 80K)$, and Technicians $(50K \sim 70K)$, we can take advantage of this information. We can find that the managers and salesmen have large variances, which deserves more samples while the engineers and technicians categories have small variances, where small number of samples will work. By reassigning the number of samples in different categories, 32 samples for Managers, 7 samples for Engineers, 59 samples for Salesmen, and 2 samples for Technicians, the overall estimated result becomes $520,066K \pm 10,113K$, which provides us more precise estimation than simple stratified sampling.

4.2 Case Study 2: Monte Carlo Linear Solver

Applying Monte Carlo sampling to estimate solutions in linear systems is originally proposed by Ulam and von Neumann and later described by Forsythe and Leibler in [6]. Considering a linear system of

$$x = Hx + b,$$

where H is an $n \times n$ non-singular matrix, b is the given constant vector, and x is vector of unknowns. The fundamental idea of the Monte Carlo solver is to construct

Markov chains by generating random walks to statistically sample the underlying Neumann series

$$I + H + H^2 + H^3 + \cdots$$

of the linear system [11]. The transition probabilities of the random walks are defined by a transition matrix P satisfying the following transition conditions:

$$\begin{aligned} P_{ij} &\geq 0; \\ \sum_j P_{ij} &\leq 1; \\ H_{ij} \neq 0 &\rightarrow P_{ij} \neq 0, \end{aligned}$$

and the termination probability T_i at row i is defined as

$$T_i = 1 - \sum_j P_{ij}.$$

Then, a random walk starting at i_0 and terminating after k steps is defined as

$$\gamma_k : r_0 \rightarrow r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_k$$

where the integers $r_0, r_1, r_2, \cdots, r_k$ are the row indices of matrix H visited during the random walk.

As noted in existing literature [13], if the necessary and sufficient condition for the convergence of Monte Carlo solver holds, a random variable $X(\gamma_k)$ defined as

$$X(\gamma_k) = \frac{H_{r_0 r_1} H_{r_1 r_2} \cdots H_{r_{k-1} r_k}}{P_{r_0 r_1} P_{r_1 r_2} \cdots P_{r_{k-1} r_k}} b_{r_k} / T_{r_k}.$$

is an unbiased estimator of component x_{r_0} in the unknown vector x.

Variance reductions have been commonly employed in the Monte Carlo algorithms to improve the sampling efficiency. Let's consider a simple linear with $H = \begin{bmatrix} 0.1 & 0.45 & 0.225 \\ -0.15 & 0.1 & -0.3 \\ -0.18 & 0.36 & 0.1 \end{bmatrix}$ and $b = \begin{bmatrix} 0.225 \\ 1.35 \\ 0.72 \end{bmatrix}$. Clearly, the exact solution is $b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

Fig. 3 compares the convergence of Monte Carlo linear solvers using three sampling schemes listed in Table 1 with different transition matrices in terms of estimate value x_1. One can find that even though all of these sampling schemes have the same convergence rate of $O(N^{-1/2})$, the estimator based on Monte Carlo Almost Optimal (MAO) scheme [3], which samples the matrix elements according to their importance, yields significant smaller variance than the other two.

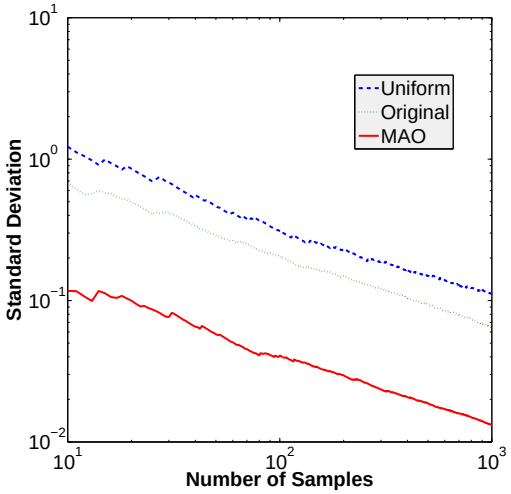


Fig. 3 Comparison of Monte Carlo Linear Solver with Different Transition Matrices

Table 1 Uniform, Original, and MAO sampling scheme with different transition matrices.

Sampling Scheme	P_{ij}	P
Uniform Sampling	$\frac{1}{n+1}$	$\begin{bmatrix} 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 \end{bmatrix}$
Original (Ulam and von Neumann)	$\lvert H_{ij}\rvert$	$\begin{bmatrix} 0.1 & 0.45 & 0.225 \\ 0.15 & 0.1 & 0.3 \\ 0.18 & 0.36 & 0.1 \end{bmatrix}$
MAO (Monte Carlo Almost Optimal)	$\frac{\lvert H_{ij}\rvert}{\sum_k \lvert H_{ik}\rvert}$	$\begin{bmatrix} 0.129032 & 0.580645 & 0.290323 \\ 0.272727 & 0.181818 & 0.545455 \\ 0.28125 & 0.5625 & 0.15625 \end{bmatrix}$

4.3 Case Study 3: Image Recovery

In this case study, we investigate the image processing technology of using a small number of pixel samples to recover an incomplete or fuzzy image. The strategy of Monte Carlo sampling plays a critical role for the quality of the recovered image.

We use an aerial image chosen from the USC-SIPI Image Database [18] as an example. The image recovery method is based on a matrix completion algorithm by optimizing the constrained nuclear norm of the matrix [2]. Two sets of pixel samples 12.19% samples in each) are used − one is generated by uniform sampling and the other by importance sampling. The importance sampling process consists of two stages:

1) initial uniform sampling is perform with 3.05% pixel samples to learn the rough pixel distribution in the image to produce a proposal function; and

2) importance sampling is performed based on the proposal function created in 1) to generate the rest 9.14% samples.

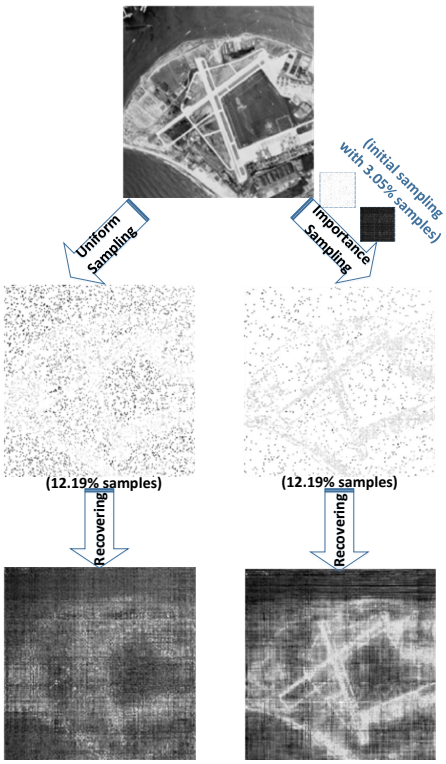


Fig. 4 Comparison of Image Recovery from Pixel Samples Generated by Uniform Sampling and Importance Sampling

As shown in Fig. 4, the pixel samples generated by the importance sampling scheme lead to a recovered image in significantly higher quality than the one by using uniform samples.

4.4 Case Study 4: Matrix Multiplication

In this case study, we investigate the Monte Carlo methods of approximating the product of large matrices. Let A be an $m \times n$ matrix and B be an $n \times p$ matrix, where m,n, and p are large. The Monte Carlo sampling algorithm to fast approximate the product matrix $C = AB$ using s samples is described as follows [4]:

1) Generate s random integers i_k between 1 and n with probability p_{i_k}, for $k = 1, \cdots, s$;
2) Set $M^{(k)} = A^{(i_k)} / \sqrt{s p_{i_k}}$ and $N_{(k)} = B_{(i_k)} / \sqrt{s p_(i_k)}$, for $k = 1, \cdots, s$;
3) Compute the matrix product of MN.

The fundamental idea of the Monte Carlo sampling algorithm is to construct a discrete random variable X with probability $p\left(X = \dfrac{A^{(k)} B_{(k)}}{p_k}\right) = p_k$ for $k = 1, \cdots, n$, where $A^{(k)}$ and $B_{(k)}$ represent the kth column of A and the kth row of B, respectively. The expectation of X is

$$E(X) = \sum_{k=1}^{n} \frac{A^{(k)} B_{(k)}}{p_k} p_k = \sum_{k=1}^{n} A^{(k)} B_{(k)},$$

which is identical to matrix C. Therefore, matrix C can be fast approximated from the product of MN, where $MN = \dfrac{1}{s} \sum_{k=1}^{s} \dfrac{A^{(i_k)} B_{(i_k)}}{p_{i_k}}$ is an estimator. The variance on each element in MN is [4]

$$Var\left[(MN)_{ij}\right] = \sum_{k=1}^{n} \frac{A_{ik}^2 A_{kj}^2}{s^2 p_k} - \frac{1}{s^2} (AB)_{ij}^2 .$$

To improve the accuracy of approximating matrix-matrix multiplication, importance sampling can be effectively applied by using the optimal probabilities

$$p_k = \frac{\left|A^{(k)}\right| \left|B_{(k)}\right|}{\sum_{j=1}^{n} \left|A^{(j)}\right| \left|B_{(j)}\right|},$$

where the expectation of approximation error can be theoretically minimized [4, 5].

Let us consider a simple toy example where

$$A = \begin{bmatrix} 1.6254 & 8.5799 & 8.5596 & 5.4832 & 9.1047 & 1.3797 \\ 9.8717 & 4.5759 & 7.5434 & 1.1230 & 8.4098 & 8.3874 \\ 4.8246 & 3.9563 & 5.1322 & 1.0143 & 8.6938 & 7.7045 \\ 0.9104 & 545.3701 & 4.6611 & 769.1278 & 9.1947 & 4.4734 \\ 14.9880 & 6.4549 & 197.7756 & 5.2276 & 88.9237 & 7.4417 \end{bmatrix}$$

and

$$B = \begin{bmatrix} 5.5899 & 2.6152 & 7.8102 & 8.2298 \\ 5.9020 & 666.2519 & 2.6396 & 7.7184 \\ 7.3150 & 143.9785 & 9.0632 & 4.9668 \\ 2.1493 & 695.7888 & 3.2657 & 974.7106 \\ 1.3544 & 758.2126 & 2.7377 & 4.6348 \\ 9.1003 & 8.3607 & 6.7709 & 27.5470 \end{bmatrix}$$

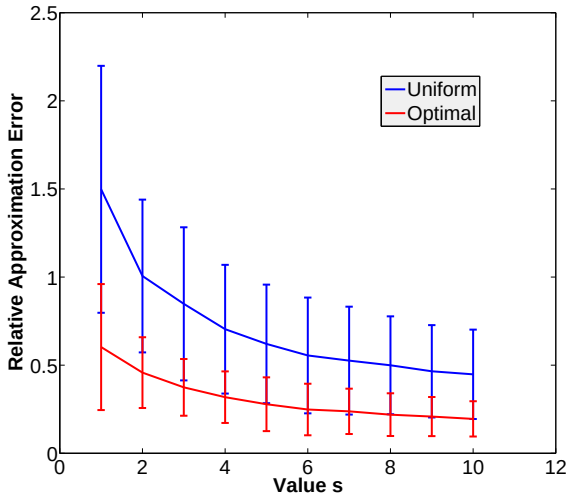


Fig. 5 Comparison of Relative Approximation Error in Matrix-Matrix Multiplication using Uniform Sampling and Importance Sampling with respect to Sample Size in 1,000 Monte Carlo runs.

Fig. 5 compares the relative approximation error of the resulting product matrix using uniform sampling and the one using important sampling with respect to sample size in $1,000$ runs. One can clearly find that when optimal selection probabilities are used, importance sampling outperforms uniform sampling with better approximation of the matrix product.

4.5 Case Study 5: Low-rank Approximation

Given a matrix A, it is often desirable to find a good low-rank approximation to A in many data analysis applications. Denote $u_1, \cdots, u_n$ and $v_1, \cdots, v_n$ as the left and right singular vectors, respectively, $\sigma_1, \sigma_2, \cdots, \sigma_n$, are singular values in non-increasing order, the matrix A can be expressed as

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i^T$$

The best k-rank approximation [7] to a matrix A is formed as $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ with minor error

$$\min_{rank(B)=k} \|A-B\|_2 = \|A-A_k\|_2 = \sigma_{k+1}.$$

To handle the computational challenges involved in very big matrices, randomized Singular Value Decomposition (SVD) algorithm with Gaussian sampling [8, 17, 12] is widely used to approximate top-k singular values and singular vectors. The rationale is to construct a small condensed subspace by sampling A, where the dominant actions of A could be fast estimated from this small subspace with relatively low computation cost and high confidence. The procedure of randomized SVD is described as follows:

1) Sample A with a standard Gaussian matrix Ω so that $Y = A\Omega$;
2) Construct a basis Q for the range of Y;
3) Compute matrix multiplication of $B = A^T Q$;
4) Perform a deterministic SVD decomposition on $B = U_B \Sigma_B V_B^T$;
5) Assign $u_j = Q v_{B_j}$, $\sigma_j = \sigma_{B_j}$, and $v_j = u_{B_j}$, $j = 1, \cdots, k$.

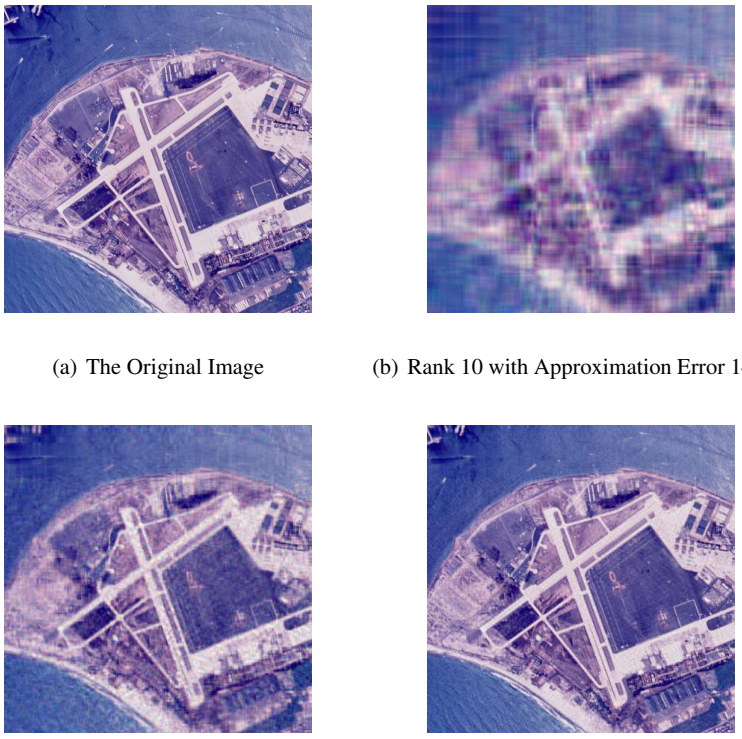
We consider an example of applying the randomized SVD algorithm to obtain a low-rank approximation of an image while minimizing the approximation error. The control variates approach is applied. First of all, the whole range space of matrix A is sampled to approximate the largest r singular components and then derive an approximate $\tilde{A}_r$ such that $\tilde{A}_r = \sum_{i=1}^{r} \sigma_i u_i v_i^T$. If the approximation error is too high, $\tilde{A}_r$ is used as the control variate to sample the next dominating singular components. This process is repeated until satisfactory approximation accuracy is achieved.

Fig. 6 shows the results of applying randomized SVD algorithm and control variate to compute a low-rank approximation of the aerial image with less than 1% error. Fig. 6(a) presents the original image. Figures 6(b) to 6(d) illustrate the adaptive reconstructed images with increasing numbers of singular components. Finally with 100 singular components, a low-rank approximation of the original image with 0.9% error is obtained.

5 Summary

Monte Carlo methods are powerful tools to analyze large data sets, particularly in the big data era when the data growth outpaces the computer processing capability. We review the basic of Monte Carlo methods and the popular techniques for variance reduction in this article. Several application examples using Monte Carlo for data analysis are presented. The rule of thumb in Monte Carlo is, the more known knowledge is incorporated into the estimator, the more uncertainty can be reduced and the better data analysis accuracy can be obtained.

Acknowledgements This work is partially supported by NSF grant 1066471 for Yaohang Li and Hao Ji acknowledges support from ODU Modeling and Simulation Fellowship.



(a) The Original Image (b) Rank 10 with Approximation Error 14.3%

(c) Rank 40 with Approximation Error 2.1% (d) Rank 100 with Approximation Error 0.9%

Fig. 6 Low-rank approximation using Randomized SVD with Gaussian sampling and control variates strategy

References

1. Burden, R., Faires, J.: Numerical Analysis. Brooks/Cole, Cengage Learning (2011)
2. Cai, J.F., Candès, E.J., Shen, Z.: A singular value thresholding algorithm for matrix completion. SIAM Journal on Optimization **20**(4), 1956–1982 (2010)
3. Dimov, I., Philippe, B., Karaivanova, A., Weihrauch, C.: Robustness and applicability of markov chain monte carlo algorithms for eigenvalue problems. Applied Mathematical Modelling **32**(8), 1511–1529 (2008)
4. Drineas, P., Kannan, R., Mahoney, M.W.: Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. SIAM Journal on Computing **36**(1), 132–157 (2006)
5. Eriksson-Bique, S., Solbrig, M., Stefanelli, M., Warkentin, S., Abbey, R., Ipsen, I.C.: Importance sampling for a monte carlo matrix multiplication algorithm, with application to information retrieval. SIAM Journal on Scientific Computing **33**(4), 1689–1706 (2011)
6. Forsythe, G.E., Leibler, R.A.: Matrix inversion by a monte carlo method. Mathematics of Computation **4**(31), 127–129 (1950)
7. Golub, G.H., Van Loan, C.F.: Matrix computations, vol. 3. JHU Press (2012)

8. Halko, N., Martinsson, P.G., Tropp, J.A.: Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM review **53**(2), 217–288 (2011)
9. Hammersley, J.M., Handscomb, D.C.: Monte carlo methods, vol. 1. Methuen London (1964)
10. Hilbert, M., López, P.: The worlds technological capacity to store, communicate, and compute information. science **332**(6025), 60–65 (2011)
11. Ji, H., Li, Y.: Reusing random walks in monte carlo methods for linear systems. Procedia Computer Science **9**, 383–392 (2012)
12. Ji, H., Li, Y.: Gpu accelerated randomized singular value decomposition and its application in image compression. In: Proceedings of Modeling, Simulation, and Visualization Student Capstone Conference, Sulfolk, 2014. (2014)
13. Ji, H., Mascagni, M., Li, Y.: Convergence analysis of markov chain monte carlo linear solvers using ulam–von neumann algorithm. SIAM Journal on Numerical Analysis **51**(4), 2107–2122 (2013)
14. Li, Y., Mascagni, M.: Grid-based monte carlo application. In: Grid ComputingGRID 2002, pp. 13–24. Springer (2002)
15. Li, Y., Mascagni, M.: Analysis of large-scale grid-based monte carlo applications. International Journal of High Performance Computing Applications **17**(4), 369–382 (2003)
16. Liu, J.S.: Monte Carlo strategies in scientific computing. Springer Science & Business Media (2008)
17. Mahoney, M.W.: Randomized algorithms for matrices and data. Foundations and Trends in Machine Learning **3**(2), 123–224 (2011)
18. SIPI, U.: The usc-sipi image database. http://sipi.usc.edu/database/