Spoofing Large Probability Mass Functions to Improve Sampling Times and Reduce Memory Costs

Jon Parker
Georgetown Unversity

Hans Engler
Georgetown University

Abstract

Sampling from a probability mass function (PMF) has many applications in modern computing. This paper presents a novel lossy compression method intended for large ($O(10^5)$) dense PMFs that speeds up the sampling process and guarantees high fidelity sampling. This compression method closely approximates an input PMF P with another PMF Q that is easy to store and sample from. All samples are drawn from Q as opposed to the original input distribution P. We say that Q "spoofs" P while this switch is difficult to detect with a statistical test. The lifetime of Q is the sample size required to detect the switch from P to Q. We show how to compute a single PMF's lifetime and present numeric examples demonstrating compression rates ranging from 62% to 75% when the input PMF is not sorted and 88% to 99% when the input is already sorted. These examples have speed ups ranging from 1.47 to 2.82 compared to binary search sampling.

1 INTRODUCTION

Sampling from a probability mass function (PMF) has many applications in modern computing. Unfortunately, PMF sampling can become somewhat cumbersome when the input distribution is not sparse and the number of states becomes large (on the order of 10^5). The problem is that the memory required to store the PMF increases linearly with respect to the sample space's size while the time required to determine the next sample may also increase (depending on the sampling implementation). This paper introduces a lossy PMF compression method that simultaneously compresses a PMF and improves sampling times by replacing the input PMF P with a close approximation Q that is easy to store and sample from. This lossy compression method has multiple desirable properties:

1. Sampling on the compressed PMF occurs faster than directly sampling the uncompressed PMF with a binary search based method.
2. The number of samples required to detect the lossy compression's error (i.e., the approximation's lifetime) can be estimated.
3. The quality of the approximation is a freely adjustable parameter.
4. Compression rates of about 75% are expected when the input PMF is unsorted while compression rates as high as 88%-99% can be obtained when the input PMF is already sorted.
5. Probabilities in the original PMF that were zero remain zero in the compressed PMF while probabilities that were non-zero remain non-zero.

This paper is an initial step towards a more ambitious goal of creating a similar compression method that simultaneously compresses a large dense stochastic matrix, enables fast sampling upon it, and guarantees that the Markov chain based on the compressed matrix is a suitable replacement for the Markov chain based on the uncompressed matrix. This goal remains elusive because there are multiple reasonable definitions for "suitable replacement" and each definition requires a different treatment.

Matlab and Java implementation source code is available at: //github.com/CenterForAdvancedModeling/

2 RELATED WORK

Quickly sampling from a finite PMF is a well-studied computational task. One standard solution is to precompute the cumulative mass function (CMF) of the

Appearing in Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS) 2014, Reykjavik, Iceland. JMLR: W&CP volume 33. Copyright 2014 by the authors.

input PMF and then execute a binary search in the CMF table. Generating samples this way is a computationally efficient $O(\log\ n)$ process but requires a CMF lookup table that is the same size as the input PMF. Chan and Asau (1974) published a method that speeds up typical binary search based sampling by storing an additional lookup table of hints beyond the precomputed CMF table. The alias method proposed by Walker (1974, 1977) guarantees $O(1)$ performance but requires two additional lookup tables each equal in size to the CMF table. These methods improve sampling speed but do nothing to assist, and some actively hinder, data compression.

Vector compression techniques are also well-studied. Consequently, there are many compression methods beyond the obvious one of using a sparse representation when appropriate. In general, lossless compression techniques like LZ compression (Ziv and Lempel, 1978) and Huffman encoding (Huffman, 1952) hinder sampling speed because they do not "unpack" quickly. On the other hand, faster lossy compression techniques like the FFT and wavelets are not expressly designed to compress PMFs so they do not include guidance about how a PMF may be deformed by the corresponding lossy compression.

To the best of our knowledge no prior work exists that simultaneously improves sampling speed, compresses a probability mass function and guarantees high fidelity sampling.

3 CLOSENESS OF DISTRIBUTION AND LIFETIME

In this section, we give a formal framework for the problem of approximating one probability distribution by another for a finite number of samples. The main result is the derivation of a distance measure of two distributions (see eq. (2)). It is then shown that it is difficult to distinguish the approximating distribution from the original one if the sample size is proportional to the reciprocal of this distance measure (Corollary 3.1).

3.1 A Single Distribution

Let P and Q be two discrete probability distributions on the same finite sample space $S = \{1, \ldots, N\}$. For a given positive integer n, consider the set S^n of all finite sequences $\{1, \ldots, N\}^n$ of length n from the sample space S. We equip S^n with the product probability measure induced either by P or Q. This is the correct probability model for drawing n independent samples from S, according to either distribution P or Q.

Definition 3.1: We say that Q spoofs P for a lifetime of L random samples given an error tolerance $\alpha \in (0, 0.25)$ when:
Given a random sample of size L drawn from Q, any hypothesis test with significance level $1-\alpha$ comparing the null hypothesis "the sample comes from P" against the alternative hypothesis "the sample comes from Q" will reject the null hypothesis with probability ≤ 0.5.

In statistical language, this says that L is a sample size such that any such test with significance level $1-\alpha$ has power at most 1/2. Note, requiring α to be small (e.g. ≤ 0.25) is typical. Whether or not a probability distribution Q spoofs another distribution P will depend on the lifetime L, the parameter α, and on some measure of closeness between P and Q.

The decision problem as it has been set up here is precisely the situation of the Neyman-Pearson Lemma (Casella and Berger, 2002). It tells us that the best way to decide between two simple hypotheses is based on the likelihood ratio. Suppose the sample is $Y = (y(1), y(2), \ldots, y(n))$ and each outcome i from the sample space S is observed x_i times. The random quantity $(x_1, \ldots, x_N)$ has a multinomial distribution. The likelihood ratio for this sample is

$$\Lambda = \frac{q_1^{x_1} q_2^{x_2} \cdots q_N^{x_N}}{p_1^{x_1} p_2^{x_2} \cdots p_N^{x_N}} = \prod_{i=1}^{N} \left(\frac{q_i}{p_i}\right)^{x_i} \tag{1}$$

We take the logarithm of this ratio for algebraic convenience and get the Log Likelihood Ratio (LLR):

$$LLR = \sum_{i=1}^{N} x_i \log \frac{q_i}{p_i} = \sum_{j=1}^{n} \log \frac{q_{y(j)}}{p_{y(j)}} = \sum_{j=1}^{n} Z_j \,.$$

The LLR is therefore a sum of n i.i.d. terms $Z_j = \log(q_{y(j)}/p_{y(j)})$. We are interested in the distribution of the LLR, in the limit of large samples that are drawn from P or from Q. Define ε_i as $\varepsilon_i = q_i/p_i - 1$, with the implicit assumption that $p_i = 0$ implies that also $q_i = 0$. Also define the quantities

$$\lambda = \frac{1}{2} \sum_{i=1}^{N} p_i \varepsilon_i^2, \quad \tilde{\varepsilon} = \max_i |\varepsilon_i| \,. \tag{2}$$

Both λ and $\tilde{\varepsilon}$ are measures for the distance between the two distributions P and Q. The quantity λ is a special type of f-divergence (Liese-Vajda, 2006) known as Pearsons χ^2-divergence. It is a second order approximation of the Kullback-Leibler divergence, defined below. The quantity $1+\tilde{\varepsilon}$ is the maximum of the Radon-Nikodym derivative dQ/dP.

Proposition 3.1: If samples are obtained from P, then for large n and small $\tilde{\varepsilon}$,

$$LLR|P \sim N(-n\lambda(1 + O(\tilde{\varepsilon}))\,,\, 2n\lambda(1 + O(\tilde{\varepsilon})).$$

If samples are obtained from Q then in the same limit

$$LLR|Q \sim \; N(n\lambda(1+O(\tilde{\varepsilon}))\,,\,2n\lambda(1+O(\tilde{\varepsilon})).$$

Proof: By the Central Limit Theorem, the LLR has an approximate normal distribution when samples are drawn from P or Q. Thus, we need only compute the mean and variance for each normal distribution. Note that $Z_j = \log(1+\epsilon_{y(j)})$. Then

$$E(Z_j|P) = \sum_{i=1}^{N} p_i \log \frac{q_i}{p_i} = -D_{KL}(P||Q) \qquad (3)$$

and similarly $E(Z_j|Q = D_{KL}(P||Q)$, where D_{KL} denotes the Kullback-Leibler (1951) divergence of two discrete probability distributions. Now using Taylor expansions of the logarithm, it is possible to show that for sufficiently small $\tilde{\varepsilon}$, up to terms of order $\tilde{\varepsilon}\lambda$

$$E(Z_j|P) \approx -\lambda, \qquad E(Z_j|Q) \approx \lambda \qquad (4)$$
$$E(Z_j^2|P) \approx 2\lambda, \qquad E(Z_j^2|Q) \approx 2\lambda\,. \qquad (5)$$

Therefore $var(Z_j|P) \approx 2\lambda \approx var(Z_j|Q)$, since $\lambda = O(\tilde{\varepsilon}^2)$. Since the LLR is a sum of n independent realizations of the Z_j, the statement follows.

Corollary 3.1: Given an error tolerance of α and two probability distributions P and Q with sufficiently small $\tilde{\varepsilon}$, then Q spoofs P for a lifetime L where

$$L = \frac{\left(\Phi^{-1}(1-\alpha)\right)^2}{2\lambda} \qquad (6)$$

and Φ^{-1} is the inverse cumulative distribution function of a standard normal distribution.

This formula is derived by observing that the best test with significance level $1-\alpha$ against the null hypothesis that sampling is done from P rejects the null if $(LLR + n\lambda)/\sqrt{2n\lambda} > \Phi^{-1}(1-\alpha)$, due to the asymptotic distribution of LLR in this case. Now if sampling is done from Q, then the probability of rejection turns out to be approximately

$$1 - \Phi\left(\Phi^{-1}(1-\alpha) - \sqrt{2n\lambda}\right)$$

due to the asymptotic distribution of LLR in this case. This quantity is bounded from above by 1/2 if $\Phi^{-1}(1-\alpha) - \sqrt{2n\lambda} \geq 0$. Solving for $n = L$ leads to the condition in Eq. (6).

4 FINDING A PIECEWISE LINEAR APPROXIMATION Q

4.1 Presorting Input

Closely approximating all entries in an arbitrary large PMF with a small number of lines is not feasible. However, closely approximating the entries of a PMF containing a non-increasing sequence of probabilities with a small number of lines is feasible. The spoofing method described herein takes advantage of this difference by requiring that the input distribution P is first sorted in descending order. All p_i terms equal to zero are removed after this sort. We store the original order in an array of integers and jetison the p_i values as we now rely on the approximating function instead. This combination of sorting and approximating saves about 75% memory when the input PMF is unsorted (assuming that 64 bit floating point numbers are replaced with 16 bit integers). If the PMF is already sorted, the memory savings are much higher, since no integers need to be stored. For the remainder of the paper it will therefore be assumed that the sequence of probabilities given by P is non-increasing and positive.

4.2 Why Piecewise Linear?

We want an approximation that compresses the input distribution, introduces little error, and enables speedy sampling. Building a piecewise linear approximation enables these three goals to be easily met. Memory savings are realized because each line in the approximation is defined by just two numbers (slope and height) but may interpolate many p_i values. Reaching an acceptable approximation error is straight forward because the approximation can be built from a variable number of lines. And finally, sampling quickly is enabled by a simple "q_i stacking" trick.

4.3 Terminology

An input PMF's non-increasing sequence of p_i terms is partitioned into one or more mutually exclusive contiguous subsequences. Each mutually exclusive contiguous subsequence is placed within a dedicated "bin". We use the symbol B_j for a generic bin. All p_i terms within such a bin are approximated by a single line. A bin's approximating line is used to compute exactly one q_i term for each p_i term inside the bin. The complete approximation Q is formed by combining all the q_i terms from all bins. As equation (2) shows, λ can be computed by summing over all (p_i, q_i) pairs. Bin B_j's contribution to λ can then be written as

$$\lambda_j = \frac{1}{2} \sum_{i \in B_j} p_i \left(\frac{q_i}{p_i} - 1\right)^2 \qquad (7)$$

and therefore $\lambda = \sum_j \lambda_j$, where the sum is over all bins B_j. Notice, bins are indexed by j and outcomes are indexed by i.

4.4 Sketch of the Approach

We want to find the "best" set of k bins with linear approximations leading to a distribution Q such that λ and k are both "small". This will allow Q to spoof P for a lifetime of L where $L = O(1/\lambda)$ and the bin storage requirements are $O(k)$. The problem appears to be intractable in this general form. We therefore propose a greedy algorithm that increases the number of bins until λ is below the required λ_L. In Figure 1, we give a rough outline of this algorithm. The remaining subsections contain details of this method that are not shown in Figure 1.

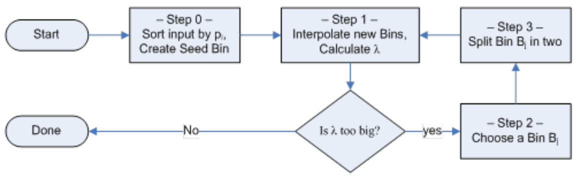
Start
– Step 0 – Sort input by p_i, Create Seed Bin
– Step 1 – Interpolate new Bins, Calculate λ
– Step 3 – Split Bin B_j in two
Done
No
Is λ too big?
yes
– Step 2 – Choose a Bin B_j

Figure 1: Simplified Flow Chart

This method starts with a preliminary sorting step and then cycles through three core steps that are repeated as necessary.

0. Sort the p_i in descending order while maintaining knowledge of the original indices. Create the initial bin using the sorted data.
1. Compute a linear approximation and λ_j for each bin without a previously computed approximation. Compute $\lambda = \sum_j \lambda_j$.
2. If λ is too large, choose a bin with $\lambda_j > 0$.
3. Split the selected bin in two and go to step 1.

Step 1 can be carried out rigorously while steps 2 and 3 implement simple but effective heuristics.

4.5 Linear Approximation of a Single Bin

In this section we cover how to find the best possible (i.e., λ_j minimizing) linear interpolation of a single bin B_j. Equation (8) specifies the line on which all q_i within a single bin will fall.

$$q_i = \alpha + \beta \left(i - \frac{n+1}{2} \right) \tag{8}$$

Here, we use n to denote the number of entries within bin B_j, index bin entries from 1 (largest p_i) to n (smallest p_i), and suppress a j index on α and β to simplify notation.

We make the additional requirement that within any given bin:

$$\sum_{i \in B_j} p_i = \sum_{i \in B_j} q_i \tag{9}$$

This requirement drastically simplifies finding the α and β that minimize λ_j because it makes the problem completely local to the current bin B_j. Without this assumption bins are related in a needlessly complicated way.

Finding α is simple because it must be set according to equation (10) so that equation (9) holds.

$$\alpha = \frac{1}{n} \sum_{i \in B_j} p_i \tag{10}$$

Directly computing α drastically simplifies finding the value of β that minimizes λ_j because (7) can now be rewritten as:

$$\lambda_j = C_0 + C_1 \beta + C_2 \beta^2 \tag{11}$$

where each C_i can be written in terms of n, α, and the p_is. The vertex of this parabala gives both the minimal λ_j and the optimal β. It is worth noting, that it is possible to compute these two important quantities without computing all the $p_i(q_i/p_i - 1)^2$ terms for each entry in the current bin. Note: It is possible that the q_i increase across a bin boundary even if the p_i are non-increasing.

4.5.1 Non-Positive q_i Values

Generating negative probability estimates is clearly not acceptable. However, the methods described above do not preclude negative q_i values. Moreover, these methods are invalid if a q_i is negative. The best way to handle this problem is to proceed using equations (10) and (11) to compute α and β. Then compute q_n, the smallest q_i. If q_n is negative set β to a value slightly larger than

$$\beta_0 = -\frac{2\alpha}{n-1} \tag{12}$$

If β were equal to β_0 the smallest q_i would be zero. Slightly increasing β above β_0 produces positive q_is, thus correcting the problem and satisfying the requirement that all q_i are positive. Once β is set, the bin's λ_j is computed and the algorithm proceeds normally. Notice, this correction obtains the same result as applying constrained optimization on β but simplifies finding the best value because the vertex of the parabala defined by (11) gives the solution the vast majority of the time.

4.6 Choosing Which Bin B_j to Split

We choose the Bin B_j for which λ_j is maximal. This seems to be the reasonable choice.

4.7 Deciding Where to Split Bin B_j

Suppose bin B_j has been selected for splitting. This bin's contribution λ_j to λ will be replaced by the sum of two terms λ'_j and λ''_j, one for each "child" bin. In principle, one could split B_j at the index where the resulting $\lambda'_j + \lambda''_j$ is minimal. Splitting B_j at this index provides the smallest possible λ when one split to bin B_j is allowed. The split location can be found with a suitable search method that requires $O(\log n)$ trials.

Unfortunately, a series of $k-1$ such optimal splits usually does not produce an optimal partition of k bins. This is because, as in any greedy approach, the efficiency of future splits is unaccounted for when optimizing the placement of the current split in isolation. Since this approach is clearly suboptimal and more sophisticated optimizations will be costly to compute we simply split a bin approximately in half, while ensuring the bin with larger p_i terms has an even number of entries. This ensures the child bin with larger p_i terms can be split again to further reduce λ when necessary.

A moment's reflection will show that this bin splitting policy guarantees that: (1) there can only be one bin that cannot be split again (ignoring perfectly approximated bins with exactly two entries) and (2) when the unsplittable bin exists it will contain the 3 smallest p_i terms. This fact is desirable because unsplittable bins with large p_i terms lock in large and unimprovable λ_j values. All successful splitting rules must ensure unsplittable bins have acceptable λ_j values given the global target λ.

4.8 A Detailed Example

To illustrate the method, a simple PMF with $N = 37$ entries was generated and approximated with another piecewise linear PMF. Figure 2 shows the original PMF at the top, the sorted PMF together with a two bin approximation in the middle, and the sorted PMF together with a three bin approximation at the bottom.

4.9 Sampling from a Piecewise Linear Approximation

We now give an efficient sampling algorithm for a piecewise linear probability distribution Q. Sampling from Q requires a two-step process. The first step selects a random bin from the set of all bins with the correct probability. The second step returns an entry from this bin, with probabilities proportional to the q_i for the i in this bin.

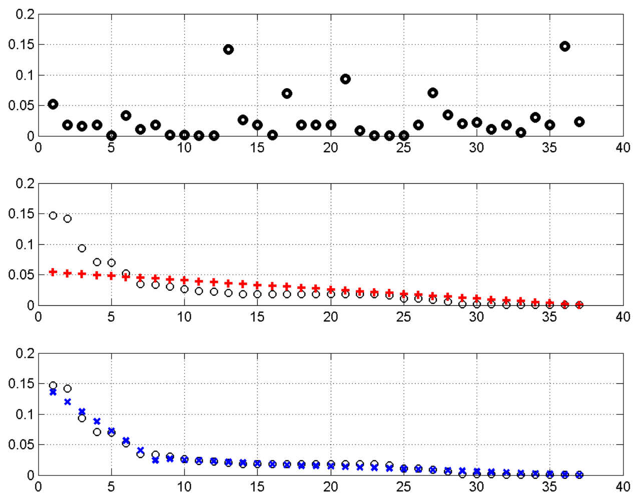

Figure 2: Two piecewise linear approximations of a PMF. Top: Unsorted PMF. Middle: Sorted PMF and approximating PMF for one bin, $\lambda = .29$. Bottom: Sorted PMF and approximating PMF for three bins, $\lambda = .045$.

4.9.1 Selecting a Random Bin

Selecting a random bin is a problem that plays to the strengths of Chan and Asau's technique. Since we expect to have relatively few bins a Chan and Asau implementation using one hint per bin ensures high-speed bin sampling and uses little memory. Thus, we begin by pre-computing the bin specific CMF and hint table required by Chan and Asau's method.

4.9.2 Selecting a Random Entry From Within a Bin

Having selected a random bin using Chan and Asau's method, we now have to select and return a random sample from within that bin. Imagine that the q_i terms within a bin are paired, the first with the last, the second with the second to last, and so on. The combined probabilities result in a set of "stacks" as shown in the right half of Figure 3. Assuming n is even, all the stacks have the same probability and a random stack can be selected easily from the two q_i terms forming this stack.

Here is a formal description, assuming again that the entries in the bin are labeled from 1 to n. Compute:

$$col = \lceil u \cdot \frac{n}{2} \rceil \tag{13}$$

where u is another $U(0,1)$ random variate. The *col* variable identifies which pair of outcomes contains the final outcome. Finally, pick between the two q_i values in this stack q_{col} and q_{n-col} by drawing another $U(0,1)$

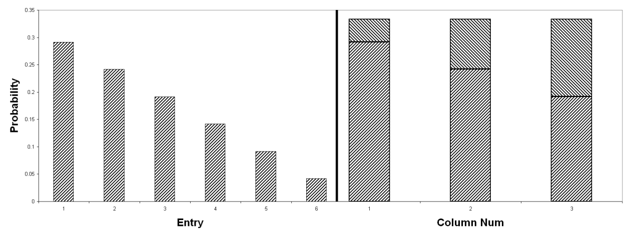

Figure 3: Stacked probabilities

random variate v or rescaling u into a $U(0,1)$ random variate v (given that u fell into the range corresponding to col) and returning:

$$sample = \begin{cases} col & \text{if } v < \frac{q_{col}}{q_{col}+q_{n-col}}; \\ n - col & otherwise. \end{cases} \tag{14}$$

A quick check shows that eq. (13) also produces a correctly distributed sample when n is odd because it is half as likely to select the stack with 2 copies of the middle q_i value as it is to select a stack with 2 distinct q_i values.

4.9.3 Sampling Shortcuts

There are two cases that allow for noteworthy shortcuts when sampling. A shortcut is available when a bin has exactly two entries. In this case, there is no need to select a random stack and we can sample one of the two entries using a modification of (14). The second shortcut is available when a bin has slope $\beta = 0$. This happens when there are many identical p_i in the original distribution. Here, simply return entry $\lceil n \cdot u \rceil$. These shortcuts become significant over the course of several several million samples (from a suitably shaped distribution) even though they only eliminate a few simple numeric operations.

4.10 Possible Improvements and Limitations

If a PMF with an even number of entries is given, our method can achieve $\lambda = 0$, simply by using bins of size 2 (which never contribute error). In this case, there is no compression. The best approximation for a distributions with an odd number of entries uses all bins of size 2 except one bin with 3 entries containing the 3 smallest p_i values. This bin will be the only source of approximation error. A user who requires a very small approximation error may therefore be better served using one of the exact methods described in Section 2. Also, a user who submits a PMF with very few entries may see little to no compression and/or speed benefits using our method.

It may be possible to improve upon the splitting strategy suggested in section 4.7. In particular, splitting a bin such that approximately half of the parent bin's probability goes to each child bin may require slightly fewer bins to reach the desired lifetime.

Splitting bins at locations found with an optimal search produces approximately the same number of bins as just splitting bins in half, at substantially increased computational cost. We speculate that other possible splitting rules will also produce roughly the same number of bins.

5 NUMERICAL EXAMPLES

We first show the effectiveness of the approximation procedure. Two PMFs on a sample space of size $n = 10^5$ were generated and sorted in descending order. The first one, P_L, is piecewise linear, with five linear pieces including a flat piece (a set of constant p_i). The second one, P_J, has probabilities proportional to $\frac{5}{i}$ for $i \le k_0 = 3333$ and proportional to $\frac{1}{i}$ for $i > k_0$. Table 1 gives the number of bins for the target λ values 10^{-2}, 10^{-4}, and 10^{-6}.

Table 1: Approximation quality for two PMFs.

PMF	Num Bins	λ
P_L	4	.0067
P_L	15	$7.8 \cdot 10^{-5}$
P_L	26	$8.5 \cdot 10^{-7}$
P_J	16	.0034
P_J	35	$7.7 \cdot 10^{-5}$
P_J	76	$9.9 \cdot 10^{-7}$

The results show that the method deals very well with piecewise linear PMFs and less well (but still quite acceptably) with a PMF with a pronounced jump. The number of bins required to approximate P_J is higher because of the jump in the otherwise smooth distribution of p_is at $i = k_0$, which requires many binary splits.

We next compare the time and memory requirements for four different sampling methods, using two different PMFs. The two distributions each come from sample spaces of size $n = 25,000$. A curved PMF, P_C, has probabilities p_i proportional to $1/u_i$ where u_i is $U(0,1)$ and a linear PMF P_S has probabilities proportional to $n + 20 - i$. The first distributions was approximated with a target $\lambda = 1.35 \cdot 10^{-8}$ using 519 bins. The second distribution is approximated perfectly ($\lambda = 0$) with a single bin. Then $N = 10^7$ samples were drawn. The time and memory required to draw these samples from each distribution are shown in Table 2.

This table shows the memory savings that are possible when using a piecewise linear approximation. In ad-

Table 2: Time (sec) and memory space (kB) required to draw 10^7 samples.

Sampling Method	P_C		P_S	
	Time	Space	Time	Space
Binary	15.1	200	21.4	200
Chan-Asau	7.2	300	7.9	300
Spoofing (unsorted)	10.3	76	11.1	50
Spoofing (sorted)	10.3	24	11.1	0.4

dition, sampling from a spoofed distribution is faster than using a binary search approach. While the approach by Chan and Asau (1974) is faster yet, that method uses more memory. The dramatic memory savings shown in the last row of Table 2 are possible because storing the input distribution's original ordering is unnecessary when it is already sorted. Since just one bin is generated when the PMF is P_S, the memory requirements in that case are minimal. In our implementation, sampling from P_C is slightly faster than sampling from P_S because the approximation of P_C contains several bins with exactly 2 entries, from which one can sample faster. We also simulated sampling from a PMF where the p_i can have only a few different values (not shown in the table). In that case, memory requirements for our implementation are similar to those for P_S, but sampling can be done about 30% faster, since sampling an entry from a given bin can be done in about half the time (see the discussion of shortcuts given above).

6 DISCUSSION: OPEN QUESTIONS AND THE STOCHASTIC MATRIX CASE

Adapting this spoofing method for large dense Markov chains, presumably by spoofing each row in the transition matrix, will enable significant memory savings and faster compute times for a widely used technique. We also know this approach will not alter important properties of a Markov chain such as its reducibility and the periodicity and recurrence of its component states. The difficulty involves computing the lifetime of the slightly altered stochastic matrix. It is unclear how to define the lifetime of such an approximation because Markov chains are used in significantly different ways.

More formally, let P be not just a single PMF, but the transition matrix of a Markov chain with stationary distribution π. Similarly, let Q be a row by row approximation of P in the spirit of this paper, with stationary distribution $\tilde{\pi}$. We would like to determine the Markov chain's lifetime, i.e. how long the chain governed by Q can be used before it can be distinguished from a chain governed by P. Unfortunately, a single definition of lifetime may not suffice because a Markov chain can be used in a variety of ways. For example, a chain may always begin at a fixed initial state. Or the chain's initial state could be drawn from a distribution (which may or may not be the stationary distribution). A user may also repeatedly restart the chain from states of his or her choosing. Each of these use cases has merit and requires a significantly different approach towards computing the lifetime. We believe the definition of a Markov chain's lifetime must reflect a specific use case and should hopefully be easy to compute. However, there is merit to defining a Markov chain's lifetime such that only the stationary distributions π and $\tilde{\pi}$ are reflected in the calculation.

There is prior work on related topics and we refer to Billingsley (1961) for statistical inference on Markov chains, where asymptotic null distributions of suitable test statistics are derived. Further fundamental results were obtained by Baum and Petrie (1966). Importantly, these prior results require that the Markov chain has already reached its stationary state. Related results on the structure of optimal strategies for detecting a switch from P to Q while the Markov chain is running (even if the chain has not yet reached stationarity) are given by Yakir (1994). We are interested in similar cases where the stationary state may not have been reached yet.

Once the Markov chain has reached stationarity, $\tilde{\pi}$ is spoofing π, but with unknown lifetime. The work by Cho and Meyer (2001) shows that $\tilde{\pi} - \pi$ is pointwise bounded in terms of $Q - P$, and a number of ways to estimate this difference are given in that reference. However, it is an open question how to bound the crucial quantity $\lambda = \sum_j |\tilde{\pi}_j - \pi_j|^2/\pi_j$ (or any other divergence of $\tilde{\pi}$ and π for that matter) effectively in terms of P and Q.

7 CONCLUSION AND OPEN QUESTIONS

The spoofing method described here provides a carefully considered compromise between sampling performance, sampling accuracy, and memory overhead. This compromise guarantees the user a requested level of accuracy while providing good speed and low memory use.

The overwhelming majority of this method's memory footprint is due to retaining the input distribution's original ordering. We made no attempt to compress this permutation. Therefore, compressing the original permutation is an obvious next step because compressing it by $X\%$ will yield a near $X\%$ improvement in overall memory use.

One last subtle benefit of the spoofing method has nothing to do with its computer science properties. The spoofing approach brings important simulation issues into sharp focus. Requiring the user to pick a tolerance level and work with samples that deviate in a controlled way from an exact version encourages him or her to consider questions such as: How stable is the output of my simulation? How good are the data this simulation is based on? How important is the error in the input data for this simulation? Clearly, it is difficult to gauge the value of considering these questions. But, intuitively, taking a moment to look at the big picture must have merit.

Acknowledgements

This work was partially supported by: the Models of Infectious Disease Agent Study (MIDAS), under Award Number U01GM070708 from the NIGMS, The Johns Hopkins Medical School DHS Center on Preparedness and Catastrophic Event Response (PACER), under Award Number N00014-06-1-0991 from the Office of Naval Research, and Joshua M. Epstein's NIH Director's Pioneer Award, Number DP1OD003874 from the Office of the Director, National Institutes of Health

References

BAUM, L. E. AND PETRIE, T. 1966. Statistical inference for probabilistic functions of finite state Markov chains. Ann. Math. Stat. Vol. 37 No. 6, 1554 - 1563.

BILLINGHSLEY, P. 1961. Statistical methods in Markov chains. Ann. Math. Stat. Vol. 32 No. 1, 12 - 40.

CASELLA, G. AND BERGER R.L. 2002. Statistical Inference. Duxbury, Pacific Grove.

CHAN, H. C. AND ASAU, Y. 1974. On generating random variates from an empirical distribution. IIE Transactions, Vol. 6 No. 2, 163 - 166.

CHO, G.E. AND MEYER, C.D. 2001. Comparison of perturbation bounds for the stationary distribution of a Markov chain. Linear Algebra Appl. Vol. 225, 137 - 150.

HUFFMAN, D.A. 1952. A method for the construction of minimum-redundancy codes. IRE Proceedings Vol. 40 No. 9, 1098 - 1101.

KULLBACK, S. AND LEIBLER, R.A. 1951. On information and sufficiency. Ann. Math. Stat. Vol. 22 No. 1, 79 - 86.

LIESE, F. AND VAJDA, I. 2006. On divergences and informations in statistics and information theory. IEEE Transactions on Information Theory Vol. 52, No. 10, 4394 - 4412.

WALKER, A.J. 1974. New fast method for generating discrete random numbers with arbitrary frequency distributions. Electronics Lett. 10, 553 - 554.

WALKER, A.J. 1977. An efficient method for generating discrete random variables with general distributions. ACM Transactions Math. Software, Vol. 3, 253 - 256.

ZIV, J. AND LEMPEL, A. 1978. Compression of individual sequences via variable-rate coding. IEEE Transactions on Information Theory Vol. 24, No. 5, 530 - 536.