An Experimental Survey of Weighted Sampling Without Replacement

with an application to Survey Sampling

Thomas Ahle

July 2022

1 Introduction

For many applications it is beneficial to be able to sample from a set S of vectors (e.g. $S \subseteq \mathbb{R}^d$) with a distribution other than uniform. Sometimes this is done to increase the amount of information in a sample, but still to estimate the same expected value $\sum_{i\in[n]} S_i/n$.

Importance sampling is the idea of rescaling the sampled values by the new probability distribution in such a way that the mean stays fixed. The downside is that this works only for sampling a single element, while for Machine Learning applications (and many other) we are interested in getting a "batch" of samples of a fixed size. This can of course be done by sampling repeatedly with replacement, but to maximize information we prefer no repetitions.

In this note we investigate how importance sampling can be made to work in this setting, and we show experimentally that our algorithms lower the variance of samples while keeping the mean unbiased.

1.1 Preliminaries

The idea of importance sampling is the following: Assume you have the uniform distribution $\mathcal{D} = (1/n, \ldots, 1/n)$ over some set $S \subseteq \mathbb{R}, |S| = n$ with mean $\mu = \mathrm{E}_{v\sim\mathcal{D}}[v]$ and variance $\sigma^2 = \mathrm{E}_{v\sim\mathcal{D}}[(v-\mu)^2]$.

We want to change $\mathcal{D}$ to another distribution $\mathcal{D}' = (p_1, \ldots, p_n)$ with the same mean, but a smaller variance. Of course changing the sampling probabilities will change the mean, but we can "compensate" for this by multiplying each value, S_i, by $1/(np_i)$, like this:

$$\mathrm{E}_{v\sim\mathcal{D}'}[v] = \sum_{i\in[n]} p_i(S_i/(np_i)) = \sum_{i\in[n]} S_i/n = \mu. \tag{1}$$

The question is thus, "how do we most efficiently choose each p_i to minimize the variance?" If we write up the new variance

$$\sum_{i\in[n]} p_i(S_i/(np_i) - \mu)^2 = \sum_{i\in[n]} p_i^{-1} S_i^2/n^2 - \mu^2,$$

We see that we want to minimize $\langle p^{-1}, S^2\rangle$, under the condition that p is a valid probability distribution ($p \geq 0$ and $\sum p = 1$.)

Using Lagrange multipliers and the KKT method, we can show that the optimal choice is to choose p proportionally to S, or more concretely

$$p_i = \frac{|S_i|}{\sum_j |S_j|}.$$

If we do this we get the variance[1]

$$\mathrm{Var}_{v\sim\mathcal{D}'}[v] = (\sum_i |S_i|)(\sum_{i\in[n]} |S_i|/n^2) - \mu^2 = (\sum_i |S_i|/n)^2 - \mu^2.$$

We note that this improves on the original variance, unless all values are equal, since $(\sum_i |S_i|/n)^2 \leq \sum_i |S_i|^2/n^2$ is exactly the AQ inequality.

The method and proof also generalizes to the case where each S_i is a vector, and the goal is to minimize $E[\|v-\mu\|_2^2]$. Here we get that p_i should be proportional to $\|v\|_2$ and the "variance" becomes $(\sum_i \|S_i\|_2/n)^2 - \|\mu\|_2^2$, which again improves over $\sum_i \|S_i\|_2^2/n^2 - \|\mu\|_2^2$,.

Note: In many cases we may only have an approximation to the norms $|S_i|$, but this doesn't prevent us from sampling unbiased. This is because the reweightening of eq. (1) only depends on knowing the sampling probabilities we choose. A worse approximation only results in a higher variance, not a biased sample.

2 Weighted Sampling without Replacement

In Machine Learning applications we usually work with data in batches of a fixed size, which means we want to do repeated sampling. Naturally it doesn't make sense to process the same value twice in the same batch, so we want to do sampling without replacement.

Sampling with replacement is a classical algorithmic problem, which can be done efficiently using Reservoir Sampling: (1) Sample an element uniformly at random (probability $1/n$), (2) Remove it from the pool, (3) Sample another element with uniformly at random (probability $1/(n-1)$), (4) and so on.

As we want to apply this method to importance sampling, a natural question becomes: How do we do weighted sampling without replacement? It is easy to sample a single value from some discrete probability distribution $(p_1, \ldots, p_n)$, but after removing the sampled value from the pool, the remaining probabilities no longer sum to 1.

[1]Note that if all values of S are non-negative, this is actually just zero. That is because we have transformed every value $S_i \mapsto S_i/(np_i) = \sum_j |S_i|/n$, so all values equal the mean, and we always get an exact result. In general we will be interested signed values and even vectors, so this "degenerate" case won't come up.

The natural method is simply re-weighting the remaining probabilities. Say the first value picked had probability p_i, then, in the second round, we sample each item, j, with probability $p_j/(1-p_i)$. In the third round $p_j/(1-p_i-\dots)$ and so on.

The issue becomes: In order to do importance sampling we *need to normalize the items by the probability that they get sampled*. Otherwise the estimator is no longer unbiased. (Recall eq. (1).)

2.1 Option 1: Computing the sample probability

Say we want to sample k out of n items with the reservoir method described above. If we can compute the probability that a given item is in the selected subset, we can use that to normalize the sample, and use importance sampling just the way it was originally described.

If we work out the details (see section 6.1) we get the formula

$$\Pr[i \text{ sampled}] = \sum_{S\subseteq[n],|S|<k,i\notin S} (-1)^{k-|S|-1}\binom{n-|S|-2}{k-|S|-1}\frac{p_i}{1-\sum_{j\in S}p_j}. \qquad (2)$$

This is a sum over all subsets of $[n]$ of size at most k that doesn't include i. The method works, but unfortunately we don't see a way to compute this in time much less than $\approx \binom{n}{k}$, which is clearly completely prohibitively expensive.

2.2 Option 2: Conditional probabilities

If we give up on computing the real inclusion probabilities, a natural alternative is to normalize by the re-weighted probabilities and see what happens.

Assume we are sampling values one by one with the reservoir sampling method above. Assume we have already sampled $t-1$ indices $I \subsetneq [n]$, $|I| = t-1$ and define $R = [n] \setminus I$ to be the remaining reservoir.

We will use the notation $w_S = \sum_{i\in S} w_i$ and similarly for x_S. Define $p_i = w_i/W_R$. For $i \le t$ let $\hat{x}_i$ be the ith item sampled. (Note $x_I = \sum_{i<t}\hat{x}_i$.) Similarly define $\hat{w}_i$ and $\hat{p}_i$ to be respectively the weight and sample probability for the ith sampled item.

With this notation, the expected value of the tth sample is $E[x_t|\mathcal{F}_{t-1}] = \sum_{i\in R} p_i x_i$. (Here the notation $E[\cdot|\mathcal{F}_{t-1}]$ is Martingale notation for the expectation conditioned on events before time t.) We define "the tth estimator" $\psi_t = \hat{x}_t/\hat{p}_t + x_I$, such that

$$E[\psi_t|\mathcal{F}_{t-1}] = x_R + x_S = x_{[n]} = n\mu.$$

We define the "combined estimator" $\Psi = \sum_{i\in[k]} \beta_i\psi_i$ for some set of k numbers β_i summing to 1. By linearity of expectation we get that

$$E[\Psi] = E\left[\sum_{i\in[k]}\beta_i\psi_i\right] = n\mu,$$

Since every e_i is a weighted sum of samples from x, we have that Ψ is similarly a sum of k vectors with some scaling to be determined.

As always, we want to minimize the variance of the estimator. We consider the Martingale difference

$$M_t = \beta_t(\psi_t - n\mu) + M_{t-1},$$

which has expectation $E[M_t] = 0$ by construction. The variance at time t is then

$$E[M_t^2|F_{t-1}] = M_{t-1}^2 + \beta_t^2 E[(\psi_t - n\mu)^2|\mathcal{F}_{t-1}] + 2\beta_t M_{t-1} E[\psi_t - n\mu|\mathcal{F}_{t-1}],$$

where the last term is 0 by construction. That leaves understanding

$$\begin{aligned} E[(\psi_t - n\mu)^2|\mathcal{F}_{t-1}] &= E[(\hat{x}_t/\hat{p}_t + x_I - n\mu)^2|\mathcal{F}_{t-1}] \\ &= E[(\hat{x}_t/\hat{p}_t)^2|\mathcal{F}_{t-1}] + (x_I - n\mu)^2 + 2E[\hat{x}_T/\hat{p}_t|\mathcal{F}_{t-1}](x_I - n\mu) \\ &= E[(\hat{x}_t/\hat{p}_t)^2|\mathcal{F}_{t-1}] + x_R^2 - 2x_R^2. \end{aligned}$$

By a similar argument to finding the original mean, we have

$$E[(\hat{x}_t/\hat{p}_t)^2|\mathcal{F}_{t-1}] = \sum_{i\in R} \frac{x_i^2}{p_i} = \sum_{i\in R} w_i \sum_{i\in R} \frac{x_i^2}{w_i}.$$

In summary we have shown

$$E[M_t^2|F_{t-1}] = M_{t-1}^2 + \beta_t^2 (w_R \sum_{i\in R} \frac{x_i^2}{w_i} - x_R^2).$$

In the optimal case, $w_i = |x_i|$, the second term is easy to recognize as

$$(\sum_{i\in R} |x_i|)^2 - (\sum_{i\in R} x_i)^2,$$

the variance from section 1.1. If we define this value σ_t^2, we get that the optimal way to choose β_i is proportional to σ_i^{-2}. The final variance we achieve is then

$$\langle \beta^2, \sigma^2 \rangle = \frac{n}{\sum_i \sigma^{-2}},$$

the harmonic mean of the sigmas.

In practice we don't know σ^2, since it depends on all the values x, and not just the once we have sampled or the weights. We thus choose to pick β_i proportional to just w_R^{-2}, the weights of the remaining items.

2.2.1 Historical notes

The method above was invented by Des Raj in 1956 [?]. However, he didn't put much consideration into the β values, and just picked $\beta_i = 1/k$ for each i.

In 1974 Rosen [?] considered some asymptotic results regarding Des Raj, including optimal asymptotic choices for β_i. However, the formulas are not computational, and Rosen falls back to analyzing the uniform, $1/k$, case.

Another approach was taken by Murthy (1957) and Rao [?]. The considered so called symmetrizations of the original Des Raj estimators. However, these methods are only efficient for small k, since they consider all the different orders the set might have been sampled.

The Des Raj method is also mentioned in many books on Survey Sampling, like [?] or [?], but as far as I can tell, they only consider the uniform beta case.

2.3 Method 3: Priority Sampling

In 2007, Duffield, Lund and Thorup [?] found a beautiful sampling algorithm, named Priority Sampling, which among many applications is useful for survey sampling.

The method is based on the following simple result:

Lemma 1. *Let $U_1, \ldots, U_n \sim U(0,1)$ and let T be the $k+1$th largest value from $w_1/U_1, \ldots, w_n/U_n$. Then*

$$E[1[w_i/U_i > T]\max(T, w_i)] = w_i.$$

The algorithm is simple: Take the k top xs by w_i/U_i and multiply them by $\max\{T, w_i\}$ as defined in the lemma.

By linearity of expectation, this gives an unbiased sample, and as was shown in the paper, one with very low variance.

```
Data: $w \in \mathbb{R}^n_+$
Result: $w' \in \mathbb{R}^k, S \subseteq [n]$ such that $E[\langle X_S, w' \rangle] \approx \sum X_i$
$u \sim \text{Uniform}([0,1]^n)$
$q \leftarrow w/u$
$S \leftarrow$ top-$k$ indices of $q$
$\tau \leftarrow (k+1)$th largest value of $q$
$w' \leftarrow [\max\{\tau/w_i, 1\} : i \in S]$
```
Algorithm 1: Priority Sample based on weights

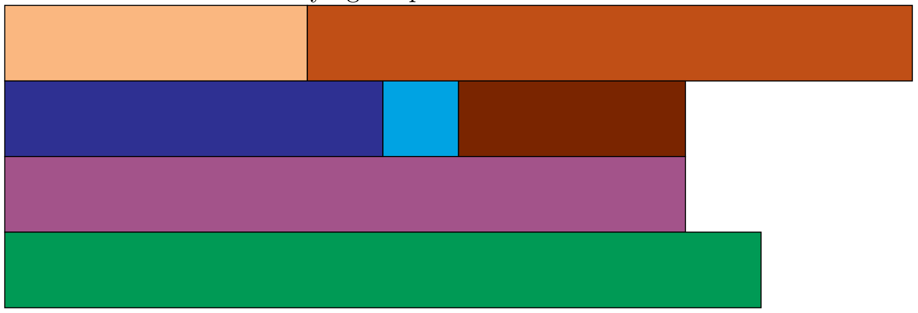
2.4 Classical Methods

Many other methods have been propoced since the middle of the last century. The most common are the following three:

Sunter's method Sunter considered the problem of "survey sampling" all the way back in 1977. It can be shown equal to another method known as Hanurav-Vijayan [?]. The idea is to do weighted sampling without replacement as long as things are going well, but switch to uniform sampling without replacement if things go bad.

Stratification (RHC) Rao, Hartley and Cochran's method [?] Idea: randomly partition in k parts of near equal size (or total weight,) then sample a single element by intended weight from each part.

We can think of it as trying to split our distributions in four rows like below:



Now we can pick a value in each row with probability proportional to its weight in the given row.

Assuming we hash each item into a row, the probability the item will get chosen is

$$E\left[\frac{w_i}{\sum_{j:h(j)=h(i)} w_j}\right] = \sum_{S\subseteq[n]\setminus\{i\}} (1/k)^{|S|}(1-1/k)^{n-1-|S|}\frac{w_i}{w_i+\sum_{j\in S} w_j},$$

which unfortunately isn't much easier to evaluate than eq. (2).

Also the method doesn't really make sense: What happens if $k=n$? Probably there'll be empty buckets. That seems bad.

The next method does better.

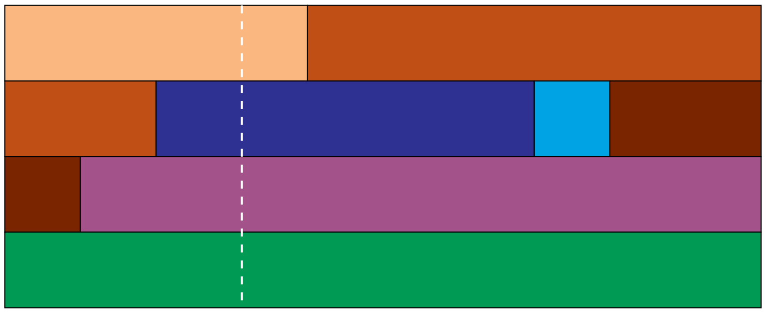
Cyclic method Idea: Make a circle of length k with n segments of length p_ik each. Pick a random value $d\in[0,1]$ and for each integer in $[0,k)$ pick the item that $d+i$ lands on.

Interesting point: If we actually try to optimize the variance given the requirement that $\sum_i p_i = k$ and $p_i\in[0,1]$, we get that we should simply always sample the large items, and re-scale the rest in the obvious way. The solution seems to be: Just make the largest values 1, and redistribute the weight. Omfg, so simple. Combined with the cyclic method, this is really, really good.

First studied by: Goodman [?] and Horvitz and Thompson [?]. Tiles book attributes it to Madow (1949). In Tiles book they complain about many joint sampling proabilities being 0, but we don't really care. We just want low variance.

In our case it's actually really nice, since we don't have to care about the order of the items, because of the way we have scaled things.

Say we have probabilities $(0.4, 0.8, 0.5, 0.1, 0.3, 0.9, 1.0)$ summing to 4, and we want to pick 4 items without replacement. We make bars of length equal to each probability and wrap them around as so:



We then pick a uniform random number in $u \sim [0,1]$, and pick the four elements intersecting the dashed line (here representing $u = 0.3134$.)[2]

It is easy to see that each item has chance equal to its probability of being sampled. The variance was analyzed in detail by by Rao and Hartley in [?] for the case where the probabilities are shuffled before creating the table.

However, since we are interested in minimizing the variance, it turns out that our choice of setting probability in proportion the the weight of each item is exactly right, since it makes the sum of weights constant independent of u, which minimized the integral

$$\int_0^1 \sum_{\text{element } i \text{ intersects } u} w_i^2 du$$

There are a bunch more methods that can match an exact probability distribution, such as Tille's nice splitting method. However, this cyclic method is basically all we

Rejection Sampling If we sample k items with replacement, and

$$\Pr[i \text{ in sample}|\text{nothing sampled twice}] = \frac{\sum_{S\subseteq[n],|S|=k,i\in S} \prod_{p\in S} p}{\sum_{S\subseteq[n],|S|=k} \prod_{p\in S} p}$$

This can be computed fast using FFT. In particular we can compute all n inclusion probabilities in $O(n(\log k)^2)$ time. The problem is that actually sampling from this distribution is hard. Well, if p is close to uniform, it actually should be pretty fast. Something like $\exp(k^2/n)$ time. But the more unbalanced it gets, the worse. It's also strange because you have to specify your probabilities summing to one. Where we are used to having probs summing to k...

For Poisson sampling we get

$$\frac{\sum_{S\subseteq[n],|S|=k,i\in S} \prod_{p\in S} p \prod_{p\notin S} 1-p}{\sum_{S\subseteq[n],|S|=k} \prod_{p\in S} p \prod_{p\notin S} 1-p}.$$

This can again be computed fast in $O(n(\log k)^2)$ time.

[2] If we need to do more than one sample, we can use the "alias method" [?] to build a simple data-structure on each row.

We could try to compute another set of probabilities $q_1, q_2, \ldots$, in a way that gives the same probabilities as above, but has the right expectation. This would make the rejection sampling method succeed much more often, which then would speed it up. Such probabilities always exists (it seems), but unfortunately they are not easy to compute. Well, maybe they are already summing to k. That's my normal setting after all.

So we can actually use that concentration result by Jakob and Anders, right?! They say that basically

$$\Pr[S = \mu] = \frac{1}{\sqrt{2\pi}\sigma} \pm c/\sigma^2,$$

where $\sigma = \sum_i p_i(1 - p_i)$. So the expected number of tries is $O(\sigma) = O(\sqrt{n})$. Each try takes n time, so the total running time is $O(n\sqrt{n})$ time. Still too slow.

3 Experiments

To test the variance in practice, we sampled sets of $n = 100$ vectors with iid. entries from different distributions (see plots below.) For each vector we computed the norm and added some noise. The 100 norms were then given to the survey sampling algorithm together with the value k of samples were were interested in. The survey sampling algorithm returns the indices to sample (compute) and the weights given to the samples. We then computed the estimated mean and it's mean squared error from the real mean. All experiments were repeated 10^6 times to make the plots readable.

In addition to the methods described above, we added two baselines:

uniform (w.r.) Simple uniform sampling of k elements with replacement.

uniform (w.o.r.) Uniform sampling of k elements without replacement.

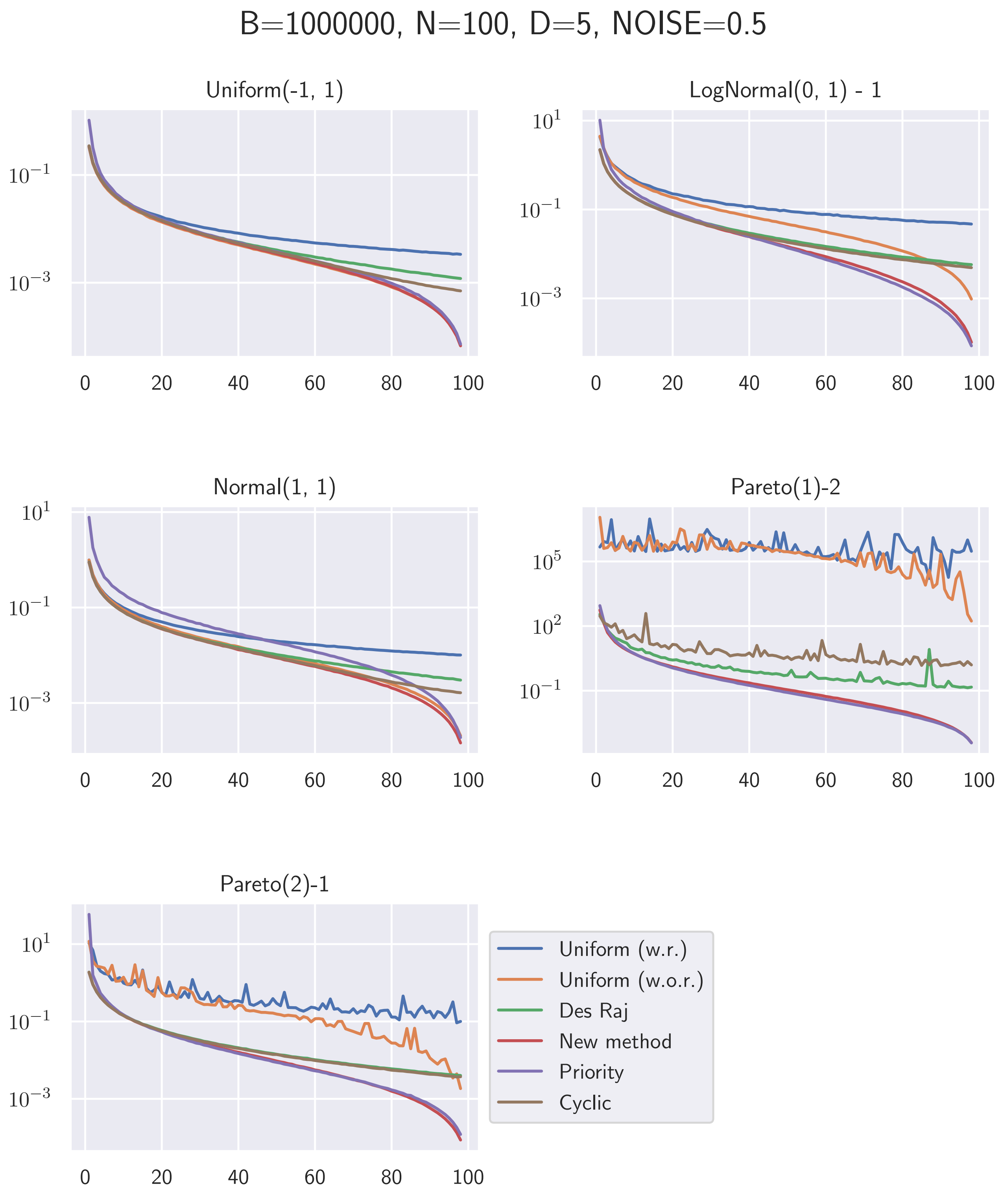


Figure 1: **Mean Squared Error of estimated means with different sample methods.** As expected, simple uniform sampling didn't do very well compared to Importance Sampling. (Exponential and Priority.) We also see that sampling with replacement is not much different from sampling without replacement with the (sub-)sample size is small, but as we start getting closer to the size of S, there is a significant difference.

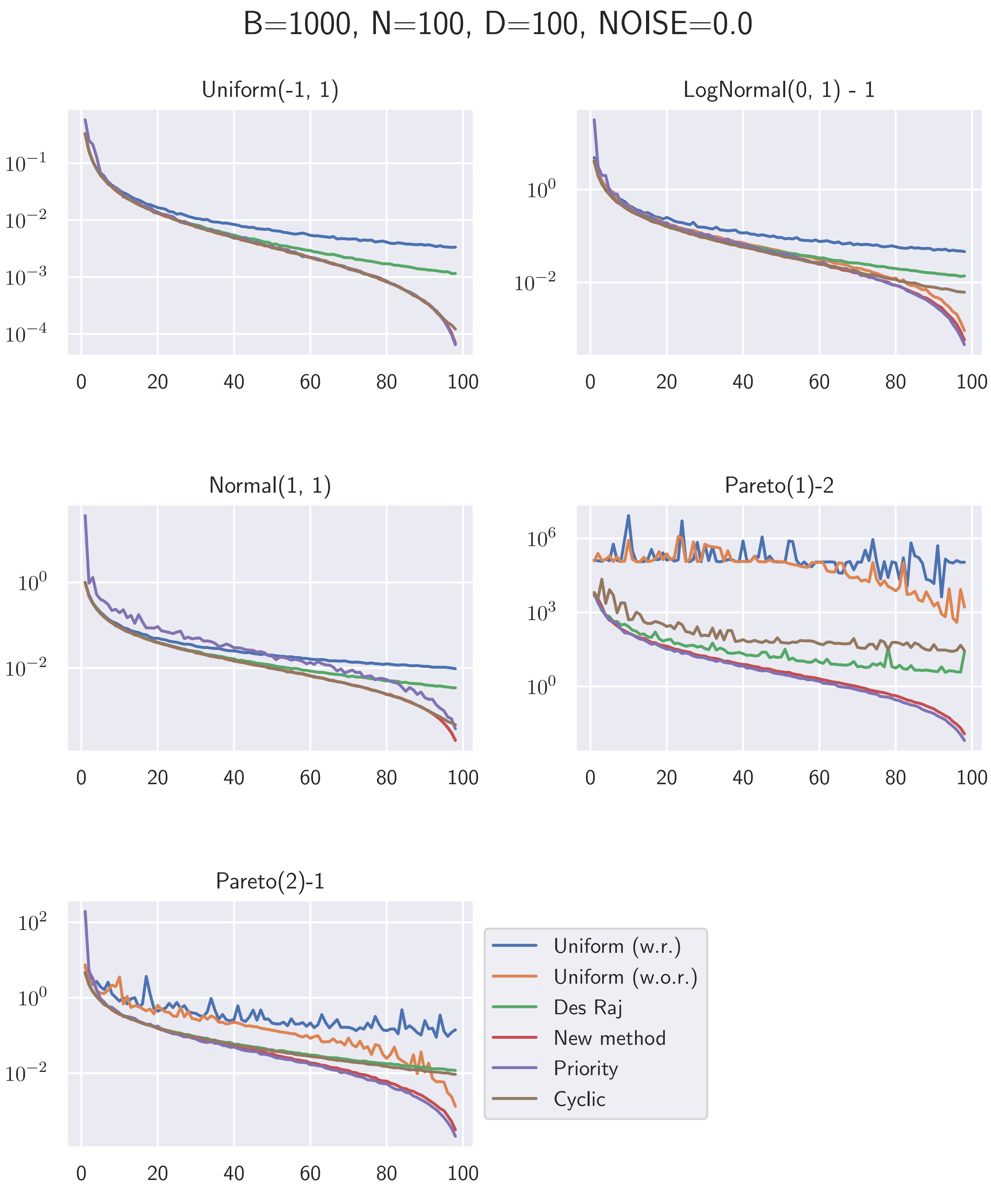


Figure 2: For larger dimensions the norms of highly concentrated distributions like Normal and Uniform tend to concentrate. This means there is less signal for survey sampling to use. However, for less concentrated distributions like LogNormal and Pareto the difference is still large.

Uniform(-1, 1)

	$k =$ 10	30	50	70	90
Uniform (w.r.)	-1.5	-2.0	-2.2	-2.3	-2.4
Uniform (w.o.r.)	-1.5	-2.1	-2.5	-2.8	-3.4
Des Raj	-1.6	-2.2	-2.5	-2.8	-2.7
New method	-1.6	**-2.2**	-2.6	-3.1	-3.6
Priority	-1.6	-2.2	**-2.7**	**-3.2**	-3.8
Sigmoid (w.o.r.)	-1.6	-2.2	-2.6	-3.1	**-4.1**
Sunter	-1.6	-2.2	-2.5	-2.9	-3.4
RHC	-1.6	-2.2	-2.6	-2.8	-3.2
Cyclic	**-1.6**	-2.2	-2.6	-3.1	-4.0

LogNormal(0, 1) - 1

	$k =$ 10	30	50	70	90
Uniform (w.r.)	-0.3	-0.8	-1.0	-1.2	-1.3
Uniform (w.o.r.)	-0.3	-1.0	-1.3	-1.7	-2.3
Des Raj	**-1.1**	-1.8	-2.2	-2.4	-2.7
New method	-1.1	-1.9	-2.4	-3.0	-3.7
Priority	-1.0	-1.9	-2.5	**-3.1**	-3.8
Sigmoid (w.o.r.)	-1.1	-1.9	-2.4	-3.0	**-4.1**
Sunter	-0.5	-1.0	-1.3	-1.7	-2.3
RHC	-1.1	-1.7	-2.0	-2.3	-2.8
Cyclic	-1.1	**-1.9**	**-2.5**	-3.1	-3.3

Normal(1, 1)

	$k =$ 10	30	50	70	90
Uniform (w.r.)	-1.0	-1.5	-1.7	-1.8	-2.0
Uniform (w.o.r.)	-1.0	-1.6	-2.0	-2.4	-3.0
Des Raj	-1.5	-2.0	-2.3	-2.6	-2.8
New method	**-1.5**	**-2.0**	**-2.4**	-2.8	-3.6
Priority	-0.9	-1.6	-2.1	-2.7	-3.6
Sigmoid (w.o.r.)	-1.4	-2.0	-2.4	**-2.8**	**-3.7**
Sunter	-1.4	-1.8	-2.0	-2.4	-2.9
RHC	-1.4	-2.0	-2.4	-2.6	-3.1
Cyclic	-1.4	-2.0	-2.4	-2.8	-3.4

Pareto(1)-2

	$k =$ 10	30	50	70	90
Uniform (w.r.)	4.6	4.6	4.5	5.4	4.6
Uniform (w.o.r.)	4.7	4.6	4.5	4.4	2.9
Des Raj	-0.0	-0.8	-1.2	-1.4	-1.7
New method	-0.1	-1.1	-1.7	-2.3	-3.3
Priority	-0.1	-1.1	-1.7	-2.2	-3.3
Sigmoid (w.o.r.)	-0.1	-1.1	-1.8	-2.4	-3.3
Sunter	4.9	5.1	4.5	4.0	2.7
RHC	0.6	-0.0	-0.7	-0.9	-1.6
Cyclic	**-0.2**	**-1.3**	**-2.0**	**-2.5**	**-3.4**

Pareto(2)-1

	$k =$ 10	30	50	70	90
Uniform (w.r.)	0.3	-0.4	-0.6	-0.8	-0.9
Uniform (w.o.r.)	-0.2	-0.5	-0.9	-1.3	-1.7
Des Raj	-1.2	-1.8	-2.2	-2.5	-2.7
New method	-1.2	-2.0	-2.5	-3.0	-3.6
Priority	-1.1	-1.9	-2.4	-2.9	-3.7
Sigmoid (w.o.r.)	-1.2	-2.0	-2.5	-3.0	-3.8
Sunter	-0.2	-0.5	-0.9	-1.2	-2.1
RHC	-1.1	-1.6	-2.0	-2.3	-2.7
Cyclic	**-1.2**	**-2.1**	**-2.7**	**-3.0**	**-3.9**

Table 1: Sets of size 100, dimension 1, noise rate 0.1, 10^6 repetitions.

Uniform(-1, 1)

	$k =$ 10	30	50	70	90
Uniform (w.r.)	-1.5	-2.0	-2.2	-2.3	-2.4
Uniform (w.o.r.)	-1.5	-2.1	-2.5	-2.8	-3.4
Des Raj	-1.5	-2.1	-2.4	-2.7	-2.9
New method	-1.5	-2.1	-2.5	-2.9	-3.5
Priority	-1.5	-2.1	-2.5	-2.9	-3.6
Sigmoid (w.o.r.)	**-1.5**	**-2.1**	**-2.5**	-2.9	-3.5
Sunter	-1.5	-2.1	-2.5	-2.8	-3.4
RHC	-1.5	-2.1	-2.5	-2.7	-3.2
Cyclic	-1.5	-2.1	-2.5	**-2.9**	**-3.6**

LogNormal(0, 1) - 1

	$k =$ 10	30	50	70	90
Uniform (w.r.)	-0.3	-0.8	-1.0	-1.2	-1.3
Uniform (w.o.r.)	-0.4	-1.0	-1.3	-1.7	-2.3
Des Raj	-0.7	-1.4	-1.7	-2.0	-2.2
New method	-0.7	-1.4	-1.9	-2.4	-3.2
Priority	-0.6	-1.4	-1.9	-2.5	-3.4
Sigmoid (w.o.r.)	-0.7	-1.4	-1.9	-2.4	-3.2
Sunter	-0.7	-1.0	-1.3	-1.7	-2.3
RHC	-0.7	-1.3	-1.7	-1.9	-2.4
Cyclic	**-0.7**	**-1.4**	**-1.9**	**-2.5**	**-3.4**

Normal(1, 1)

	$k =$ 10	30	50	70	90
Uniform (w.r.)	-1.0	-1.5	-1.7	-1.8	-2.0
Uniform (w.o.r.)	-1.0	-1.6	-2.0	-2.4	-3.0
Des Raj	-1.1	-1.7	-2.0	-2.3	-2.5
New method	**-1.1**	**-1.7**	**-2.1**	**-2.5**	-3.1
Priority	-0.7	-1.4	-1.8	-2.2	-2.9
Sigmoid (w.o.r.)	-1.1	-1.7	-2.1	-2.5	-3.1
Sunter	-1.1	-1.7	-2.0	-2.4	-3.0
RHC	-1.1	-1.7	-2.1	-2.3	-2.8
Cyclic	-1.1	-1.7	-2.1	-2.5	**-3.1**

Pareto(1)-2

	$k =$ 10	30	50	70	90
Uniform (w.r.)	6.1	6.1	6.1	6.1	6.3
Uniform (w.o.r.)	6.1	6.1	6.1	5.3	4.2
Des Raj	1.0	0.1	0.4	-0.6	-0.8
New method	0.7	-0.3	-1.0	-1.6	-2.5
Priority	0.7	-0.4	-1.1	-1.7	-2.6
Sigmoid (w.o.r.)	0.7	-0.3	-1.0	-1.7	-2.5
Sunter	6.1	6.1	6.1	5.3	6.0
RHC	2.2	1.0	0.3	0.0	-0.5
Cyclic	**0.6**	**-0.4**	**-1.1**	**-1.8**	**-2.8**

Pareto(2)-1

	$k =$ 10	30	50	70	90
Uniform (w.r.)	-0.0	-0.5	-0.7	-0.8	-0.9
Uniform (w.o.r.)	-0.0	-0.6	-1.0	-1.3	-2.0
Des Raj	-0.9	-1.5	-1.9	-2.1	-2.4
New method	-0.9	-1.6	-2.1	-2.6	-3.3
Priority	-0.8	-1.6	-2.1	-2.6	-3.3
Sigmoid (w.o.r.)	-0.9	-1.6	-2.1	-2.6	-3.4
Sunter	0.1	-0.6	-1.0	-1.4	-1.9
RHC	-0.8	-1.4	-1.7	-2.0	-2.4
Cyclic	**-0.9**	**-1.7**	**-2.2**	**-2.8**	**-3.5**

Table 2: Sets of size 100, dimension 5, noise rate 0.01, 10^6 repetitions.

4 Conclusion

Importance sampling is a great method, and it can relatively easily be made to work with sampling without replacement. For the fastest methods (priority and Des Raj) there is little downside unless your noise is very large / advice is very bad.

Most of the time priority sampling provided the best MSE for a given sample size. However, the modified Des Raj method was never far behind, and in the highly biased case of Normal(1,1) priority sampling didn't do well.

5 Future work

Another interesting possibility is the case where we not only know an approximation to the norms of the items we are interested in sampling, but also know an approximation to the pairwise inner products.

A useful case to consider is a set of vectors resulting from a bimodal distribution. Assume the modes are a and b, and all vectors are either sampled from $N(a, \varepsilon)$ or $N(b, \varepsilon)$. The most informative sample of $k = 2$ vectors might be taking one expected to be from the first batch, and one expected to be from the second. However, sampling this way is not possible when only knowing the approximate norms.

References

6 Appendix

6.1 Computing Inclusion Probabilities

If we want to generate exactly k samples from each batch, we can take the bottom k indices from $[X_i w_i : i \in [n]]$, where $X_i \sim \mathrm{Exp}(1)$. For $k = 1$ this gives the expected probability $w_k / \sum_i w_i$, and for $k = n$ it takes everything. The only issue is that it's not easy to compute the exact probability that a given sample is included, which may bias the results.

$$\begin{aligned}
\Pr[o(X_i) \le k] &= \int_{\mathbb{R}^+} \sum_{S\subseteq[n],|S|<k,i\notin S} \left(\prod_{j\in S}(1-e^{-t\lambda_j})\right)\left(\lambda_i e^{-t\lambda_i}\right)\left(\prod_{j\notin S,j\neq i} e^{-t\lambda_j}\right) dt \\
&= \lambda_i \int \sum_{S\subseteq[n],|S|<k,i\notin S} \sum_{T\subseteq S} (-1)^{|T|} \exp(-t\sum_{j\in T}\lambda_j)\exp(-t\sum_{j\notin S}\lambda_j)dt \\
&= \lambda_i \sum_{T\subseteq S\subseteq[n],|S|k,i\notin S} \frac{(-1)^{|T|}}{\sum_{j\in([n]\setminus S)\cup T}\lambda_j} \\
&= \lambda_i \sum_{S'\subseteq[n],|S'|<k,i\notin S} \frac{\sum_{t<k-s}(-1)^t\binom{n-s-1}{t}}{\sum_{j\notin S'}\lambda_j} \\
&= \lambda_i \sum_{S'\subseteq[n],|S'|<k,i\notin S} \frac{(-1)^{k-|S'|-1}\binom{n-|S'|-2}{k-|S'|-1}}{\sum_{j\notin S'}\lambda_j} \\
&= \lambda_i \sum_{S'\subseteq[n],|S'|<k,i\notin S} \frac{\binom{k-n}{k-|S'|-1}}{\sum_{j\notin S'}\lambda_j}
\end{aligned}$$

So clearly it is possible to compute the bottom-k probability, but perhaps not much faster than $\binom{n}{k}$ time, which isn't great.