Near-Optimal Averaging Samplers

Zhiyang Xun∗
Department of Computer Science
University of Texas at Austin
zxun@cs.utexas.edu

David Zuckerman†
Department of Computer Science
University of Texas at Austin
diz@utexas.edu

May 31, 2024

Abstract

We present the first efficient averaging sampler that achieves asymptotically optimal randomness complexity and near-optimal sample complexity for natural parameter choices. Specifically, for any constant $\alpha > 0$, for $\delta > 2^{-\text{poly}(1/\varepsilon)}$, it uses $m + O(\log(1/\delta))$ random bits to output $t = O(\log(1/\delta)/\varepsilon^{2+\alpha})$ samples $Z_1, \ldots, Z_t \in \{0, 1\}^m$ such that for any function $f : \{0, 1\}^m \to [0, 1]$,

$$\Pr \left[ \left| \frac{1}{t} \sum_{i=1}^{t} f(Z_i) - \mathbb{E} f \right| \leq \varepsilon \right] \geq 1 - \delta.$$ 

The sample complexity is optimal up to the $O(\varepsilon^n)$ factor.

We use known connections with randomness extractors and list-decodable codes to give applications to these objects.

1 Introduction

Randomization plays a crucial role in computer science, offering significant benefits across various applications. However, obtaining true randomness can be challenging. It’s therefore natural to study whether we can achieve the benefits of randomization while using few random bits.

One of the most basic uses of randomness is sampling. Given oracle access to an arbitrary function $f : \{0, 1\}^m \to [0, 1]$ on a large domain, our goal is to estimate its average value. By drawing $t = O(\log(1/\delta)/\varepsilon^2)$ independent random samples $Z_1, \ldots, Z_t \in \{0, 1\}^m$, the Chernoff bound guarantees that the average value $\left| \frac{1}{t} \sum_{i=1}^{t} f(Z_i) - \mathbb{E} f \right| \leq \varepsilon$ with probability at least $1 - \delta$. This method uses full independence in sampling, but we can also pursue more efficient strategies. This leads to the following definition:

Definition 1.1 ([BR94]). A function $\text{Samp} : \{0, 1\}^n \to (\{0, 1\}^m)^t$ is a $(\delta, \varepsilon)$ averaging sampler with $t$ samples using $n$ random bits if for every function $f : \{0, 1\}^m \to [0, 1]$, we have

$$\Pr_{(Z_1, \ldots, Z_t) \sim \text{Samp}(U_n)} \left[ \left| \frac{1}{t} \sum_{i=1}^{t} f(Z_i) - \mathbb{E} f \right| \leq \varepsilon \right] \geq 1 - \delta.$$ 

*Supported by NSF award CCF-2008868 and the NSF AI Institute for Foundations of Machine Learning (IFML).
†Supported by NSF Grant CCF-2312573 and a Simons Investigator Award (#409864).
We would like to construct explicit samplers using few random bits that have sample complexity close to the optimal. Researchers have made progress towards this goal, and a summary is given in Table 1. Bellare and Rompel [BR94] suggested that interesting choices of parameters are $\varepsilon = 1/\text{poly}(m)$ and $\delta = \exp(-\text{poly}(m))$. This enables us to use $\text{poly}(m)$ random bits and generate $\text{poly}(m)$ samples.

<table>
<thead>
<tr>
<th>Due to</th>
<th>Method</th>
<th>Random Bits</th>
<th>Sample Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CEG95]</td>
<td>Lower Bound</td>
<td>$m + \log(1/\delta) - \log(O(t))$</td>
<td>$\Omega(\log(1/\delta)/\varepsilon^2)$</td>
</tr>
<tr>
<td>CEG95</td>
<td>Non-Explicit</td>
<td>$m + 2\log(2/\delta) + \log(1/\varepsilon)$</td>
<td>$2\log(4/\delta)/\varepsilon^2$</td>
</tr>
<tr>
<td>Standard</td>
<td>Full Independence</td>
<td>$O(m \log(1/\delta)/\varepsilon^2)$</td>
<td>$O(\log(1/\delta)/\varepsilon^2)$</td>
</tr>
<tr>
<td>[CG89]</td>
<td>Pairwise Independence</td>
<td>$2m$</td>
<td>$O(1/(\delta \varepsilon^2))$</td>
</tr>
<tr>
<td>[Gil98]</td>
<td>Expander Walks</td>
<td>$m + O(\log(1/\delta)/\varepsilon^2)$</td>
<td>$O(\log(1/\delta)/\varepsilon^2)$</td>
</tr>
<tr>
<td>[BR94]</td>
<td>Iterated Sampling</td>
<td>$O(m + (\log m \log(1/\delta))$</td>
<td>$\text{poly}(1/\varepsilon, \log(1/\delta), \log m)$</td>
</tr>
<tr>
<td>[Zuc97]</td>
<td>Hash-Based Extractors</td>
<td>$(1 + \alpha)(m + \log(1/\delta))$</td>
<td>$\text{poly}(1/\varepsilon, \log(1/\delta), m)$</td>
</tr>
<tr>
<td>RVW00</td>
<td>Zig-Zag Extractors</td>
<td>$m + (1 + \alpha) \log(1/\delta)$</td>
<td>$\text{poly}(1/\varepsilon, \log(1/\delta))$</td>
</tr>
<tr>
<td>Here</td>
<td>[RVW00] + Almost $\ell$-wise Uniform</td>
<td>$m + O(\log(1/\delta))$</td>
<td>$O(\log(1/\delta)/\varepsilon^{2+\alpha})$</td>
</tr>
</tbody>
</table>

Table 1: Comparison of averaging samplers, $\alpha$ any positive constant, $\varepsilon = 1/\text{poly}(m)$, and $\delta = \exp(-\text{poly}(m))$.

The best existing randomness-efficient averaging sampler comes from the equivalence between averaging samplers and extractors [Zuc97]. Improving Zuckerman’s construction, Reingold, Vadhan, and Wigderson [RVW00] gave a $(\delta, \varepsilon)$ averaging sampler for domain $\{0,1\}^m$ that uses $m + (1 + \alpha) \log(1/\delta)$ random bits for any positive constant $\alpha$. This almost matches the lower bound in [CEG95]. However, a notable gap remains in sample complexity: the existing construction’s complexity $\text{poly}(1/\varepsilon, \log(1/\delta))$ does not align with the optimal $O(\log(1/\delta)/\varepsilon^2)$. This raised an open problem: Can we design an averaging sampler that not only meets the $O(m + \log(1/\delta))$ random bit requirement but also achieves the more efficient sample complexity of $O(\log(1/\delta)/\varepsilon^2)$ [BR94, Zuc97, Gol11]?

We note that such algorithms do exist for general samplers, which queries $f$ and computes the estimation of $\mathbb{E} f$ by an arbitrary computation [BGG93]. However, many applications require the use of averaging samplers, such as the original use in interactive proofs [BR94]. Beyond these applications, averaging samplers act as a fundamental combinatorial object that relate to other notions such as randomness extractors, expander graphs, and list-decodable codes [Zuc97, Vad07].

1.1 Our Sampler

In this paper, we construct a polynomial-time computable $(\delta, \varepsilon)$ averaging sampler $\text{Samp}$ with near-optimal sample complexity using an asymptotically optimal number of random bits. In fact, the sampler we constructed is a strong sampler, defined as follows:

**Definition 1.2.** A $(\delta, \varepsilon)$ averaging sampler $\text{Samp}$ is strong if for every sequence of $t$ functions $f_1, \ldots, f_t : \{0,1\}^m \rightarrow [0,1]$, we have

$$\Pr_{(Z_1, \ldots, Z_t) \sim \text{Samp}(U_m)} \left[ \frac{1}{t} \sum_i (f_i(Z_i) - \mathbb{E} f_i) \right] \leq \varepsilon \geq 1 - \delta.$$ 

We then state our main theorem:
Theorem 1. For every constant $\alpha > 0$, there exists an efficient strong $(\delta, \varepsilon)$ averaging sampler for domain $\{0, 1\}^m$ with $O(\frac{1}{\varepsilon^2 + \alpha} \log \frac{1}{\delta})$ samples using $m + O((1 + \frac{\log \log(1/\delta)}{\log(1/\varepsilon)}) \log \frac{1}{\delta})$ random bits.

We have the next immediate corollary.

**Corollary 2.** For arbitrary positive constants $\alpha$ and $C$, given any $(\delta, \varepsilon)$ such that $\log(1/\delta) < \varepsilon^{-C}$, there exists an explicit $(\delta, \varepsilon)$ averaging sampler for domain $\{0, 1\}^m$ with $O(\frac{1}{\varepsilon^2 + \alpha} \log \frac{1}{\delta})$ samples using $m + O(\log \frac{1}{\delta})$ random bits.

In particular, when $\varepsilon = 1/poly(m)$ and $\delta = \exp(-poly(m))$, our sampler achieves $O(\frac{1}{\varepsilon^2 + \alpha} \log \frac{1}{\delta})$ sample complexity while using $m + O(\log \frac{1}{\delta})$ random bits, which is optimal up to the $\varepsilon^\alpha$ factor.

### 1.2 Randomness Extractors

Our sampler construction has implications for randomness extractors. A randomness extractor is a function that extracts almost-uniform bits from a low-quality source of randomness. We define the quality of a random source as its min-entropy.

**Definition 1.3.** The min-entropy of a random variable $X$ is

$$H_\infty(X) := \min_{x \in \text{supp}(X)} \log \left( \frac{1}{\Pr[X = x]} \right).$$

An $(n, k)$-source is a random variable on $n$ bits with min-entropy at least $k$.

Then a randomness extractor is defined as:

**Definition 1.4 ([NZ96]).** A function $\text{Ext} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ is a $(k, \varepsilon)$ extractor if for every $(n, k)$-source $X$, the distribution $\text{Ext}(X, U_d) \approx_\varepsilon U_m$. We say $\text{Ext}$ is a strong $(k, \varepsilon)$ extractor if for every $(n, k)$-source $X$, the distribution $(\text{Ext}(X, Y), Y) \approx_\varepsilon U_{m+d}$, where $Y$ is chosen from $U_d$.

Randomness extractors are used in many areas within theoretical computer science. However, there has been little study of explicit extractors with the right dependence on $\varepsilon$. This is a particular concern in cryptography, where $\varepsilon$ is often very small. Existentially, there are extractors with seed length $d = \log(n - k) + 2\log(1/\varepsilon) + O(1)$, and there is a matching lower bound [RT00].

Zuckerman [Zuc97] showed that averaging samplers are essentially equivalent to extractors. Specifically, an extractor $\text{Ext} : \{0, 1\}^n \times \{2^d\} \rightarrow \{0, 1\}^m$ can be seen as a sampler that generates $\text{Ext}(X, i)$ as its $i$-th sample point using the random source $X$. Using this equivalence, we can show that our sampler implies an extractor with almost optimal dependence on $\varepsilon$.

**Theorem 3.** For any positive constants $C$ and $\alpha$, all $\varepsilon \geq 0$, and all $k$ such that $k \geq n - \varepsilon^{-C}$, there exists an efficient strong $(k, \varepsilon)$ extractor $\text{Ext} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ with $m = \Omega(k)$ and $d = \log(n - k) + (2 + \alpha)\log(1/\varepsilon) + O(1)$.

Prior to our work, extractors with a seed length dependence on $\varepsilon$ achieving $2\log(1/\varepsilon)$ or close to it were based on the leftover hash lemma [BBR88, IZ89, HILL99] and expander random walks [Gil98, Zuc07]. Extractors using the leftover hash lemma have a seed length of $n + 2\log(1/\varepsilon)$, which is far from optimal. Expander random walks give a $(k, \varepsilon)$ extractor with $k > (1 - \Omega(\varepsilon^2))n$ and an optimal seed length of $\log(n - k) + 2\log(1/\varepsilon) + O(1)$. Our extractor allows for smaller $k$ whenever $\varepsilon < 1/n^c$ for an arbitrarily small constant $c > 0$. 

3
1.3 Techniques

Our construction is very simple, and is based on two observations:

1. Rather than querying every sample point produced by a sampler \(\text{Samp}\), we can use a second sampler \(\text{Samp}'\) to pick certain samples for querying. This reduces the sample complexity because the number of queried samples just depends on \(\text{Samp}'\). Since the domain of \(\text{Samp}'\) is much smaller than the original domain, this allows more efficient sampling strategies. This observation has been utilized in previous sampler constructions [BR94, Gol11].

2. The bottleneck of generating an almost \(\ell\)-wise uniform sequence over a large domain \(\{0, 1\}^m\) lies in sampling \(\ell\) independent random points, which costs \(\ell m\) random bits. Since we can only afford \(O(m)\) random bits, we are restricted to generating constant-wise uniform samples. However, for a much smaller domain, we can use few random bits to generate an almost \(\ell\)-wise uniform sequence for large \(\ell\).

Our construction is outlined as follows. Let \(\text{Ext} : \{0, 1\}^n \times [t'] \to \{0, 1\}^m\) be the extractor-based sampler in [RVW00]. Let \(Y_1, \ldots, Y_t\) be an almost \(\ell\)-wise uniform sequence over domain \([t']\), thinking of \(t \ll t'\). Our sampler is then defined by

\[
\text{Samp} := (\text{Ext}(X, Y_1), \text{Ext}(X, Y_2), \ldots, \text{Ext}(X, Y_t)).
\]

In this construction, we use the almost \(\ell\)-wise uniform sequence to sub-sample from the extractor-based sampler. This can be viewed as a composition, similar to other cases such as Justesen codes [Jus72] and the first PCP theorem [ALM+98], where the goal is to optimize two main parameters simultaneously by combining two simpler schemes, each optimizing one parameter without significantly compromising the other.

Previous works have also applied almost \(\ell\)-wise independence in extractor constructions. Srinivasan and Zuckerman [SZ99] proved a randomness-efficient leftover hash lemma by sampling an almost \(\ell\)-wise independent function using uniform seeds and inputting a weak random source. Our construction inverts this process: we generate an \(\ell\)-wise uniform sequence using a weak random source and then choose an index uniformly. Furthermore, Ran Raz’s two-source extractor [Raz05] utilized two weak random sources to sample an almost \(\ell\)-wise uniform sequence and an index separately. This is a more general construction, but if we directly apply Raz’s error bound in our analysis Lemma 3.3, the final sample complexity will be off by a \(\log(1/\delta)\) factor.

It might be of independent interest to readers who are familiar with the Nisan-Zuckerman pseudorandom generator [NZ96]. Our sampler has the same structure as the Nisan-Zuckerman generator, and one can view our construction from the perspective of pseudorandom generators. In the classical analysis of the Nisan-Zuckerman generator, ensuring a success probability of \(1 - \delta\) demands an extractor error smaller than \(\delta\), since an error at any step implies a complete loss of control. However, in our setting, every output sample has a very small effect on the final answer. This enables us to use an extractor with much larger error than \(\delta\) here.

1.4 List-Decodable Codes

Another perspective on averaging samplers is its connection to error-correcting codes. Ta-Shma and Zuckerman [TZ04] showed that strong randomness extractors are equivalent to codes with good soft-decision decoding, which is related to list recovery. From this perspective, the composition scheme in our construction is similar to code concatenation.
For codes over the binary alphabet, soft decision decoding amounts to list decodability, which we focus on here. We give good list-decodable codes without using the composition. That is, by just applying our almost $\ell$-wise uniform sampler on the binary alphabet, we can get a binary list-decodable code with rate $\Omega(\varepsilon^{2+\alpha})$ and non-trivial list size, although the list size is still exponential.

**Theorem 4.** For every constant $\alpha > 0$: there exists an explicit binary code with rate $\Omega(\varepsilon^{2+\alpha})$ that is $((\rho = \frac{1}{2} - \varepsilon, L) $ list-decodable with list size $L = 2^{(1-c)n}$ for some constant $c = c(\alpha) > 0$.

Prior to our work, the best known code rate was $\Omega(\varepsilon^{3})$ by Guruswami and Rudra [GR08]. We emphasize that their code achieved a list size of $L = \text{poly}(n)$, while our list size is exponentially large, making our code unlikely to be useful.

2 Preliminaries

**Notations.** We use $[t]$ to represent set $\{1, \ldots, t\}$. For integer $m$, $U_m$ is a random variable distributed uniformly over $\{0, 1\}^m$. For random variables $X$ and $Y$, we use $X \approx_{\varepsilon} Y$ to represent the statistical distance (total variation distance) between $X$ and $Y$ is at most $\varepsilon$. We use the term “efficient” to mean polynomial-time computable.

2.1 Extractor-Based Sampler

As mentioned above, averaging samplers are equivalent to extractors. We will introduce this in detail in Section 4.1. Reingold, Vadhan, and Wigderson used this equivalence to achieve the following:

**Theorem 2.1** ([RVW00, Corollary 7.3], see also [Gol11, Theorem 6.1]). For every constant $\alpha > 0$, there exists an efficient $(\delta, \varepsilon)$ averaging sampler over $\{0, 1\}^m$ with $\text{poly}(1/\varepsilon, \log(1/\delta))$ samples using $m + (1 + \alpha) \cdot \log_2(1/\delta)$ random bits.

For ease of presentation, we often denote an extractor-based averaging sampler by $\text{Ext} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$, where $\text{Ext}(X, i)$ is the $i$-th output sample point of the sampler using randomness input $X$. Therefore, the sample complexity of $\text{Ext}$ is $2^d$.

2.2 Almost $\ell$-wise Uniform

An almost $\ell$-wise uniform sequence is a sequence of random variables such that the marginal distribution of every $\ell$ of them is close to uniform.

**Definition 2.2** ([NN93]). A sequence of random variables $Z_1, \ldots, Z_t \in \{0, 1\}^m$ is said to be $\gamma$-almost $\ell$-wise uniform if for all subsets $S \subseteq [t]$ such that $|S| \leq k$,

$$(Z_i)_{i \in [S]} \approx_{\gamma} U_{m\times|S|}.$$  

**Lemma 2.3** ([NN93]). There exists an efficient algorithm that uses $O(\ell m + \log(1/\gamma) + \log \log t)$ random bits to generate a $\gamma$-almost $\ell$-wise uniform sequence $z_1, \ldots, z_t \in \{0, 1\}^m$.

Using standard techniques, we have the following concentration bound for almost $\ell$-wise uniform sequences (see Appendix A for the proof). Bellare and Rompel [BR94] derived a similar bound for exact $\ell$-wise uniform sequences.
Lemma 2.4. Let $Z_1, \ldots, Z_t \in \{0,1\}^m$ be a sequence of $\gamma$-almost $\ell$-wise uniform variables for an even integer $\ell$. Then for every sequence of functions $f_1, \ldots, f_t : \{0,1\}^m \rightarrow [0,1]$,

$$\Pr \left[ \left| \frac{1}{t} \sum_{i=1}^{t} (f_i(Z_i) - \mathbb{E} f_i) \right| \leq \varepsilon \right] \geq 1 - \left( \frac{5\sqrt{\ell}}{\varepsilon \sqrt{t}} \right)^{\ell} - \frac{\gamma}{\varepsilon^t}.$$ 

2.3 Composition of Samplers

The idea of composing samplers has been studied before. More specifically, Goldreich proved the following proposition.

Proposition 2.5 ([Gol11]). Suppose we are given two efficient samplers:

- A $(\delta, \varepsilon)$ averaging sampler for domain $\{0,1\}^m$ with $t_1$ samples using $n_1$ random bits.
- A $(\delta', \varepsilon')$ averaging sampler for domain $\{0,1\}^{\log t_1}$ with $t_2$ samples using $n_2$ random bits.

Then, there exists an efficient $(\delta + \delta', \varepsilon + \varepsilon')$ averaging sampler for domain $\{0,1\}^m$ with $t_2$ samples using $O(n_1 + n_2)$ random bits.

3 Main Results

Our construction is based on a reduction lemma that constructs a sampler for domain $\{0,1\}^m$ based on a sampler for domain $\{0,1\}^{O(\log(1/\varepsilon) + \log \log(1/\delta))}$. We exploit the fact that when composing averaging samplers, the final sample complexity depends on only one of the samplers. Our strategy is:

- Apply the extractor sampler in Theorem 2.1 as a $(\delta/2, \varepsilon/2)$ sampler over domain $\{0,1\}^m$. This uses $m + O(\log(1/\delta))$ random bits and generates $\text{poly}(1/\varepsilon, \log(1/\delta))$ samples.
- By Proposition 2.5, we only need to design a $(\delta/2, \varepsilon/2)$ averaging sampler over domain $\{0,1\}^{O(\log(1/\varepsilon) + \log \log(1/\delta))}$ using $O(\log(1/\delta))$ random bits. The total sample complexity will be equal to the sample complexity of this sampler. For this sampler, we use almost $\ell$-wise uniformity.

To formally prove the reduction lemma, we establish the next lemma, which demonstrates the explicit composition of samplers and proves that this composition maintains the properties of a strong sampler. The proof follows from the idea of Proposition 2.5.

Lemma 3.1. Let $\text{Ext} : \{0,1\}^n \times \{0,1\}^d \rightarrow \{0,1\}^m$ be a $(\delta, \varepsilon)$ averaging sampler, and let $\text{Samp}_{\text{base}}$ be a $(\delta', \varepsilon')$ averaging sampler for domain $\{0,1\}^d$. Suppose $Y_1, \ldots, Y_t \in \{0,1\}^d$ are the samples generated by $\text{Samp}_{\text{base}}$, i.e., for uniformly random source $R$,

$$\text{Samp}_{\text{base}}(R) = (Y_1, \ldots, Y_t).$$

Then, for a uniformly random $X \in \{0,1\}^n$,

$$\text{Samp}(R, X) := (\text{Ext}(X, Y_1), \ldots, \text{Ext}(X, Y_t))$$

is a $(\delta' + \delta, \varepsilon' + \varepsilon)$ averaging sampler for domain $\{0,1\}^m$. Furthermore, if $\text{Samp}_{\text{base}}$ is strong, then $\text{Samp}$ is also a strong $(\delta' + t\delta, \varepsilon' + \varepsilon)$ averaging sampler.
Proof. We only prove the case when $\text{Samp}_{\text{base}}$ is a strong sampler, and the non-strong case follows similarly. Let $f_1, \ldots, f_t : \{0,1\}^m \rightarrow [0,1]$ be an arbitrary sequence of functions. By the definition of strong samplers, we have for every $f_i$,

$$\Pr_{X \sim U_n} \left[ \left| \mathbb{E}_{Y \sim U_d} f_i(\text{Ext}(X,Y)) - \mathbb{E} f_i \right| \leq \varepsilon \right] \geq 1 - \delta.$$ 

By a union bound over all $f_1, \ldots, f_t$, we have \footnote{Note that for the non-strong case, we don’t need a union bound here. Thus, we can save a factor of $t$ in the error parameter.}

$$\Pr_{X \sim U_n} \left[ \forall i \in [t] : \left| \mathbb{E}_{Y \sim U_d} f_i(\text{Ext}(X,Y)) - \mathbb{E} f_i \right| \leq \varepsilon \right] \geq 1 - t\delta. \tag{1}$$

For an arbitrary $x$, view $f_i(\text{Ext}(x,\cdot))$ as a Boolean function on domain $\{0,1\}^d$. Therefore, since $Y_1, \ldots, Y_t$ are generated by a strong $(\delta, \varepsilon)$ sampler,

$$\Pr_{Y_1, \ldots, Y_t} \left[ \frac{1}{t} \sum_{i=1}^{t} \left( f_i(\text{Ext}(x,Y_i)) - \mathbb{E} f_i \right) \leq \varepsilon' \right] \geq 1 - \delta'. \tag{2}$$

By the triangle inequality and a union bound over equations (1) and (2), we have

$$\Pr_{X, Y_1, \ldots, Y_t} \left[ \frac{1}{t} \sum_{i=1}^{t} \left( f_i(\text{Ext}(X,Y_i)) - \mathbb{E} f_i \right) \leq \varepsilon' + \varepsilon \right] \geq 1 - \delta' - t\delta.$$ 

This proves that $(\text{Ext}(X,Y_1), \ldots, \text{Ext}(X,Y_t))$ is a strong $(\delta' + t\delta, \varepsilon' + \varepsilon)$ averaging sampler. \hfill $\square$

Instantiating Lemma 3.1 with the extractor-based sampler from Theorem 2.1 gives:

**Lemma 3.2 (Main Reduction Lemma).** For any $\alpha > 0$: For a sufficiently large constant $C > 0$, suppose there exists an efficient $(\delta', \varepsilon')$ averaging sampler $\text{Samp}_{\text{base}}$ for domain $\{0,1\}^{C(\log(1/\varepsilon)+\log\log(1/\delta))}$ with $t$ samples using $n$ random bits. Then

- There exists an efficient $(\delta + \delta', \varepsilon + \varepsilon')$ averaging sampler $\text{Samp}$ for domain $\{0,1\}^m$ with $t$ samples using $m + (1 + \alpha)\log(1/\delta) + n$ random bits.

- If $\text{Samp}_{\text{base}}$ is strong, then there exists an efficient $(\delta + \delta', \varepsilon + \varepsilon')$ averaging sampler $\text{Samp}$ for domain $\{0,1\}^m$ with $t$ samples using $m + (1 + \alpha)\log(t/\delta) + n$ random bits.

Proof. By Theorem 2.1, there exists an explicit $(\delta/t, \varepsilon)$ averaging sampler $\text{Ext} : \{0,1\}^{n'} \times \{0,1\}^d \rightarrow \{0,1\}^m$ with $n' = m + (1 + \alpha)(\log(t/\delta))$ and $d = \log(\text{poly}(1/\varepsilon, \log(t/\delta))) \leq C(\log\log(t/\delta) + \log(1/\varepsilon))$ for some large enough constant $C$.

First, we need to verify that $\text{Samp}_{\text{base}}$ can work for domain $\{0,1\}^d$, i.e., that $d \leq C(\log(1/\varepsilon) + \log\log(1/\delta))$. Without loss of generality, we assume $\log t \leq C(\log(1/\varepsilon) + \log\log(1/\delta))$; otherwise, we can just use the trivial sampler that outputs the whole domain. Thus, $\text{Samp}_{\text{base}}$ can successfully work for $\{0,1\}^d$ since

$$d \leq C(\log(1/\varepsilon) + \log\log(1/\delta)) + \frac{C}{2} \log \log t \leq C(\log(1/\varepsilon) + \log\log(1/\delta)).$$

Next, we analyze the number of random bits that $\text{Samp}$ needs. We need $n'$ random bits to generate $x$ and we need $n$ bits to generate $y_1, \ldots, y_t$.

Therefore, the total number of random bits we need is $n' + n = m + (1 + \alpha)\log(t/\delta) + n$. The lemma then follows from Lemma 3.1. \hfill $\square$
Next, we show that for domain \{0, 1\}^m with \(m \leq O(\log(1/\varepsilon) + \log(1/\delta))\), we can use an almost \(\ell\)-wise uniform sequence to design a strong averaging sampler with near-optimal sample complexity.

**Lemma 3.3.** For any constant \(\alpha > 0\), there exists an efficient strong \((\delta, \varepsilon)\) averaging sampler for domain \{0, 1\}^m with \(O(\frac{1}{\varepsilon^{2+\alpha}} \log \frac{1}{\delta})\) samples using \(O(\frac{m \log(1/\delta)}{\log(1/\varepsilon)} + \log(1/\delta))\) random bits.

**Proof.** We begin by setting \(\ell = \frac{2 \log(2/\delta)}{\alpha \log(1/\varepsilon)}\), \(\gamma = \frac{\delta \varepsilon}{2}\), and \(t = \frac{50 \log(1/\delta)}{\alpha \varepsilon^{2+\alpha} \log(1/\varepsilon)}\). We then define our sampler by outputting a \(\gamma\)-almost \(\ell\)-wise uniform sequence \(Z_1, \ldots, Z_t \in \{0, 1\}^m\). Taking the parameters of Lemma 2.4, observe

\[
\left( \frac{5\sqrt{t}}{\varepsilon \sqrt{t}} \right)^\ell = (\varepsilon^{a/2})^\ell \frac{\delta}{\varepsilon} = \frac{1}{2},
\]

and

\[
\frac{\gamma}{\varepsilon^2} = \frac{\delta}{2}.
\]

Therefore, for every sequence of functions \(f_1, \ldots, f_t : \{0, 1\}^m \to [0, 1]\),

\[
\Pr \left[ \left| \frac{1}{\ell} \sum_{i=1}^{t} (f_i(Z_i) - \mathbb{E} f_i) \right| \leq \varepsilon \right] \geq 1 - \delta.
\]

Furthermore, Lemma 2.3 shows that we have an efficient algorithm that uses only \(O(\frac{m \log(1/\delta)}{\log(1/\varepsilon)} + \log(1/\delta))\) random bits to generate this \(\gamma\)-almost \(\ell\)-wise uniform sequence.

\(\square\)

Combining Lemma 3.2 and Lemma 3.3, we prove the main theorem.

**Proof of Theorem 1.** By Lemma 3.2, our goal is to design an efficient strong \((\delta/2, \varepsilon/2)\) averaging sampler \(\text{Samp}_{\text{base}}\) for domain \{0, 1\}^{C(\log(1/\varepsilon) + \log \log(1/\delta))} for some large enough constant \(C\). The theorem is proved if \(\text{Samp}_{\text{base}}\) generates \(O(\frac{1}{\varepsilon^{2+\alpha}} \log \frac{1}{\delta})\) samples using \(O(\frac{\log(1/\delta) \log \log(1/\delta)}{\log(1/\varepsilon)} + \log(1/\delta))\) random bits. The almost \(\ell\)-wise uniform sampler defined in Lemma 3.3 for \(m = C(\log(1/\varepsilon) + \log \log(1/\delta))\) satisfies these conditions, and proves the theorem.

**Remark 3.4.** Instead of using an almost \(\ell\)-wise uniform sequence, we can also use a perfectly \(\ell\)-wise uniform sequence to establish Theorem 1. Specifically, an \(\ell\)-wise uniform sequence would give a strong \((\delta, \varepsilon)\) averaging sampler with \(O(\frac{1}{\varepsilon^{2+\alpha}} \log \frac{1}{\delta})\) samples using \(O(\frac{m \log(1/\delta)}{\log(1/\varepsilon)} + \log(1/\delta) + \log \log(1/\delta))\) random bits. This matches the bound of the almost \(\ell\)-wise uniform sampler in Lemma 3.3 when \(m = \Theta(\log(1/\varepsilon) + \log \log(1/\delta))\). However, it performs poorly for small domains, making it unable to yield Theorem 4.

4 Applications to Extractors and Codes

4.1 Applications to Extractors

Zuckerman showed that averaging samplers are equivalent to randomness extractors [Zuc97]. Here we state the only direction that we need.

**Lemma 4.1 ([Zuc97]).** An efficient strong \((\delta, \varepsilon)\) averaging sampler \(\text{Samp} : \{0, 1\}^n \to (\{0, 1\}^m)^t\) gives an efficient strong \((n - \log(1/\delta) + \log(1/\varepsilon), 2\varepsilon)\) extractor \(\text{Ext} : \{0, 1\}^n \times \{0, 1\}^{\log t} \to \{0, 1\}^m\).

Applying Lemma 4.1 on Corollary 2 gives Theorem 3.
4.2 Application to List-Decodable Codes

Error-correcting codes are combinatorial objects that enable messages to be accurately transmitted, even when parts of the data get corrupted. Codes have been extensively studied and have proven to be extremely useful in computer science. Here we focus on the combinatorial property of list-decodability, defined below.

**Definition 4.2.** A code $ECC: \{0,1\}^n \to (\{0,1\}^m)^t$ is $(\rho, L)$ list-decodable if for every received message $r \in (\{0,1\}^m)^t$, there are at most $L$ messages $x \in \{0,1\}^n$ such that $d_H(ECC(x), r) \leq \rho t$, where $d_H$ denotes the Hamming distance. A code is binary if $m = 1$.

We focus on the binary setting, i.e., $m = 1$.

**Lemma 4.3** ([TZ04]). An efficient strong $(\delta, \varepsilon)$ averaging sampler $\text{Samp}: \{0,1\}^n \to \{0,1\}^t$ over the binary domain gives an efficient binary code that is $(\rho = \frac{1}{2} - \varepsilon, \delta 2^n)$ list-decodable with code rate $R = n/t$.

Applying Lemma 4.3 to our almost $\ell$-wise uniform sampler in Lemma 3.3 gives Theorem 4.

5 Open Problems

Our work raises interesting open problems.

- Can we remove the $\log(1/\delta) < \varepsilon^{-C}$ requirement in Corollary 2? Specifically, for constant $\varepsilon$ and exponentially small $\delta$, can we design a $(\delta, \varepsilon)$ averaging sampler with $O(\log(1/\delta))$ samples using $O(m + \log(1/\delta))$ random bits?
- Is it possible to reduce the list size of the list-decodable codes in Theorem 4 to poly($n$) by the structure of the list?
- Comparing to the sampler in [RVW00] which uses $m + (1 + \alpha)\log(1/\delta)$ random bits, our construction requires $m + O(\log(1/\delta))$ random bits. Can we improve our randomness efficiency while maintaining a good sample complexity?
- Is there a way to eliminate the $\varepsilon^\alpha$ factor in the sample complexity? For $\varepsilon = 1/\text{poly}(m)$ and $\delta = \exp(-\text{poly}(m))$, can we design an efficient averaging sampler that is asymptotically optimal in both randomness and sample complexity? This will fully answer the open question raised by Bellare and Rompel.

Acknowledgements

We thank Oded Goldreich and Dana Moshkovitz for helpful comments.

Bibliography


A Proof of Lemma 2.4

Proposition A.1 (Marcinkiewicz–Zygmund inequality [RL01]). Let \( \{X_i, i \geq 1\} \) be a sequence of independent random variables with \( \mathbb{E} X_i = 0, \mathbb{E}|X_i|^p < \infty \). Then for \( p \geq 2 \):

\[
\mathbb{E} \left| \sum_{i=1}^{n} X_i \right|^p \leq C(p)n^{p/2 - 1} \sum_{i=1}^{n} \mathbb{E}|X_i|^p,
\]

where \( C(p) \leq (3\sqrt{2})^{p}p^{p/2} \).

Proof of Lemma 2.4. Let \( W_i := f_i(Z_i) - \mathbb{E} f_i \). We have

\[
\Pr \left[ \sum_{i=1}^{t} W_i > t\epsilon \right] \leq \frac{\mathbb{E} \left| \sum_{i=1}^{t} W_i \right|}{(t\epsilon)^{\ell}}.
\]

Let \( W_1', \ldots, W_t' \) be a sequence of independent random variables where \( W_i' := f_i(U_{(0,1)^m}) - \mathbb{E} f_i \). Since the \( W_i \)'s are \( \gamma \)-almost \( \ell \)-wise independent and \( |W_i| \leq 1 \), we have

\[
\mathbb{E} \left[ \sum_{i=1}^{t} W_i \right] = \mathbb{E} \left[ \sum_{i=1}^{t} W_i' \right] \leq \mathbb{E} \left[ \sum_{i=1}^{t} W_i' \right] + \gamma t^\ell = \mathbb{E} \left[ \sum_{i=1}^{t} W_i' \right] + \gamma t^\ell.
\]

Since \( \mathbb{E} W_i' = 0 \) and \( |W_i'| \leq 1 \), they satisfy the conditions for Marcinkiewicz–Zygmund inequality. We have

\[
\mathbb{E} \left[ \sum_{i=1}^{t} W_i' \right] \leq (3\sqrt{2})^{\ell/2}t^{\ell/2} - 1 \sum_{i=1}^{t} \mathbb{E}|W_i'|^{\ell} \leq (5\sqrt{tt})^{\ell}.
\]

Therefore,

\[
\frac{\mathbb{E} \left[ \sum_{i=1}^{t} W_i' \right]}{(t\epsilon)^{\ell}} \leq \frac{(5\sqrt{tt})^{\ell}}{(t\epsilon)^{\ell}} + \frac{\gamma}{\epsilon^{\ell}}.
\]