Let's play the following game: Given a stream of Bern(p) bits, with *unknown* p, we want to turn them into pure random bits, i.e., independent fair coin flips Bern(1/2). Our goal is to find a universal way to extract the most number of bits.

In 1951 von Neumann proposed the following scheme: Divide the stream into pairs of bits, output 0 if 10, output 1 if 01, otherwise do nothing and move to the next pair. Since both 01 and 10 occur with probability pq (where $q = 1 - p$), regardless of the value of p, we obtain fair coin flips at the output. To measure the efficiency of von Neumann's scheme, note that, on average, we have $2n$ bits in and $2pqn$ bits out. So the efficiency (rate) is pq. The question is: Can we do better?

Several variations:

1. Universal v.s. non-universal: know the source distribution or not.
2. Exact v.s. approximately fair coin flips: in terms of total variation or Kullback-Leibler divergence

We only focus on the universal generation of exactly fair coins.

28.1 Setup

Recall from Lecture 6 that $\{0,1\}^* = \cup_{k\geq 0}\{0,1\}^k = \{\varnothing, 0, 1, 00, 01, \dots\}$ denotes the set of all finite-length binary strings, where $\varnothing$ denotes the empty string. For any $x \in \{0,1\}^*$, let $l(x)$ denote the length of x.

Let's first introduce the definition of random number generator formally. If the input vector is X^n, denote the output (variable-length) vector by $Y \in \{0,1\}^*$. Then the desired property of Y is the following: Conditioned on the length of Y being k, Y is uniformly distributed on $\{0,1\}^k$.

Definition 28.1 (Extractor)**.** We say $\Psi : \{0,1\}^* \to \{0,1\}^*$ an *extractor* if

1. $\Psi(x)$ is a prefix of $\Psi(y)$ if x is a prefix of y.
2. For any n and any $p \in (0,1)$, if $X^n \overset{\text{i.i.d.}}{\sim} \text{Bern}(p)$, then $\Psi(X^n) \sim \text{Bern}(1/2)^k$ conditioned on $l(\Psi(X^n)) = k$.

The *rate* of Ψ is

$$r_\Psi(p) = \limsup_{n\to\infty} \frac{\mathbb{E}[l(\Psi(X^n))]}{n}, \quad X^n \overset{\text{i.i.d.}}{\sim} \text{Bern}(p).$$

Note that the von Neumann scheme above defines a valid extractor Ψ_{vN} (with $\Psi_{\text{vN}}(x^{2n+1}) = \Psi_{\text{vN}}(x^{2n})$), whose rate is $r_{\text{vN}}(p) = pq$.

28.2 Converse

No extractor has a rate higher than the binary entropy function. The proof is simply data processing inequality for entropy and the converse holds even if the extractor is allowed to be non-universal (depending on p).

Theorem 28.1. *For any extractor Ψ and any $p \in (0,1)$,*

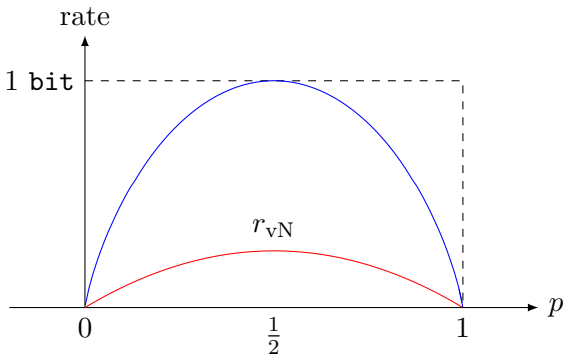
$$r_\Psi(p) \le h(p) = p \log_2 \frac{1}{p} + q \log_2 \frac{1}{q}.$$

Proof. Let $L = \Psi(X^n)$. Then

$$nh(p) = H(X^n) \ge H(\Psi(X^n)) = H(\Psi(X^n)|L) + H(L) \ge H(\Psi(X^n)|L) = \mathbb{E}[L] \quad \texttt{bits},$$

where the step follows from the assumption on Ψ that $\Psi(X^n)$ is uniform over $\{0,1\}^k$ conditioned on $L = k$. □

The rate of von Neumann extractor and the entropy bound are plotted below. Next we present two extractors, due to Elias [Eli72] and Peres [Per92] respectively, that attain the binary entropy function. (More precisely, both ideas construct a sequence of extractors whose rate approaches the entropy bound).



28.3 Elias' construction of RNG from lossless compressors

The intuition behind Elias' scheme is the following:

1. For iid X^n, the probability of each string only depends its *type*, i.e., the number of 1's. Therefore conditioned on the number of ones, X^n is uniformly distributed (over the type class). This observation holds universally for any p.

2. Given a uniformly distributed random variable on some finite set, we can easily turn it into *variable-length* fair coin flips. For example, if U is uniform over $\{1,2,3\}$, we can map $1 \mapsto \varnothing, 2 \mapsto 0$ and $3 \mapsto 1$.

Lemma 28.1. *Given U uniformly distributed on $[M]$, there exists $f : [M] \to \{0,1\}^*$ such that conditioned on $l(f(U)) = k$, $f(U)$ is uniformly over $\{0,1\}^k$. Moreover,*

$$\log_2 M - 4 \le \mathbb{E}[l(f(U))] \le \log_2 M \quad \texttt{bits}.$$

Proof. We defined f by partitioning $[M]$ into subsets whose cardinalities are powers of two, and assign elements in each subset to binary strings of that length. Formally, denote the binary expansion of M by $M = \sum_{i=0}^{n} m_i 2^i$, where the most significant bit $m_n = 1$ and $n = \lfloor \log_2 M \rfloor + 1$. Those non-zero m_i's defines a partition $[M] = \cup_{j=0}^{t} M_j$, where $|M_i| = 2^{i_j}$. Map the elements of M_j to $\{0,1\}^{i_j}$.

To prove the bound on the expected length, the upper bound follows from the same entropy argument $\log_2 M = H(U) \geq H(f(U)) \geq H(f(U)|l(f(U))) = \mathbb{E}[l(f(U))]$, and the lower bound follows from

$$\mathbb{E}[l(f(U))] = \frac{1}{M}\sum_{i=0}^{n} m_i 2^i \cdot i = n - \frac{1}{M}\sum_{i=0}^{n} m_i 2^i (n-i) \geq n - \frac{2^n}{M}\sum_{i=0}^{n} 2^{i-n}(n-i) \geq n - \frac{2^{n+1}}{M} \geq n-4,$$

where the last step follows from $n \leq \log_2 M + 1$. □

Elias' extractor. Let $w(x^n)$ define the Hamming weight (number of ones) of a binary string. Let $T_k = \{x^n \in \{0,1\}^n : w(x^n) = k\}$ define the Hamming sphere of radius k. For each $0 \leq k \leq n$, we apply the function f from Lemma 28.1 to each T_k. This defines a mapping $\Psi_{\mathrm{E}} : \{0,1\}^n \to \{0,1\}^*$ and then we extend it to $\Psi_{\mathrm{E}} : \{0,1\}^n \to \{0,1\}^*$ by applying the mapping per n-bit block and discard the last incomplete block. Then it is clear that the rate is given by $\frac{1}{n}\mathbb{E}[l(\Psi_{\mathrm{E}})(X^n)]$. By Lemma 28.1, we have

$$\mathbb{E}\log\binom{n}{w(X^n)} - 4 \leq \mathbb{E}[l(\Psi_{\mathrm{E}})(X^n)] \leq \mathbb{E}\log\binom{n}{w(X^n)}$$

Using Stirling's expansion (see, e.g., [Ash65, Lemma 4.7.1]), we have $\frac{2^{nh(p)}}{\sqrt{8npq}} \leq \binom{n}{k} \leq \frac{2^{nh(p)}}{\sqrt{2\pi npq}}$ where $p = 1 - q = k/n \in (0,1)$ and hence $\mathbb{E}[l(\Psi_{\mathrm{E}})(X^n)] = nh(p) + O(\log n)$. Therefore the extraction rate approaches $h(p)$ as $n \to \infty$.

28.4 Peres' iterated von Neumann's scheme

Main idea: Recycle the bits thrown away in von Neumann's scheme and iterate. What did von Neumann's extractor discard: (a) bits from equal pairs. (b) location of the distinct pairs. To achieve the entropy bound, we need to extract the randomness out of these two parts as well.

First some notations: Given x^{2n}, let $k = l(\Psi_{\mathrm{vN}}(x^{2n}))$ denote the number of consecutive distinct bit-pairs.

- Let $1 \leq m_1 < \ldots < m_k \leq n$ denote the locations such that $x_{2m_j} \neq x_{2m_j-1}$.
- Let $1 \leq i_1 < \ldots < i_{n-k} \leq n$ denote the locations such that $x_{2i_j} = x_{2i_j-1}$.
- $y_j = x_{2m_j}$, $v_j = x_{2i_j}$, $u_j = x_{2j} \oplus x_{2j+1}$.

Here y^k are the bits that von Neumann's scheme outputs and both v^{n-k} and u^n are discarded. Note that u^n is important because it encodes the location of the y^k and contains a lot of information. Therefore von Neumann's scheme can be improved if we can extract the randomness out of both v^{n-k} and u^n.

Peres' extractor: For each $t \in \mathbb{N}$, recursively define an extractor Ψ_t as follows:

- Set Ψ_1 to be von Neumann's extractor Ψ_{vN}, i.e., $\Psi_1(x^{2n+1}) = \Psi_1(x^{2n}) = y^k$.
- Define Ψ_t by $\Psi_t(x^{2n}) = \Psi_t(x^{2n+1}) = (\Psi_1(x^{2n}), \Psi_{t-1}(u^n), \Psi_{t-1}(v^{n-k}))$.

Example: Input $x = 100111010011$ of length $2n = 12$. Output recursively:

$$(011)(110100)(101)$$
$$(1)(010)(10)(0)$$
$$(1)(0)$$

Note that the bits that enter into the iteration are longer iid. To compute the rate of Ψ_t, it is convenient to introduce the notion of exchangeability. We say X^n are *exchangeable* if the joint distribution is invariant under permutation, that is, $P_{X_1,\dots,X_n} = P_{X_{\pi(1)},\dots,X_{\pi(n)}}$ for any permutation π on $[n]$. In particular, if X_i's are binary, then X^n are exchangeable if and only if the joint distribution only depends on the *Hamming weight*, i.e., $P_{X^n=x^n} = p(w(x^n))$. Examples: X^n is iid Bern(p); X^n is uniform over the Hamming sphere T_k.

Lemma 28.2 (Ψ_t preserves exchangebility)**.** *Let X^{2n} be exchangeable and $L = \Psi_1(X^{2n})$. Then conditioned on $L = k$, Y^k, U^n and V^{n-k} are independent and exchangeable. Furthermore, $Y^k \overset{i.i.d.}{\sim} Bern(\frac{1}{2})$ and U^n is uniform over T_k.*

Proof. If suffices to show that $\forall y, y' \in \{0,1\}^k, u, u' \in T_k$ and $v, v' \in \{0,1\}^{n-k}$ such that $w(v) = w(v')$, $\mathbb{P}[Y^k = y, U^n = u, V^{n-k} = v | L = k] = \mathbb{P}[Y^k = y', U^n = u', V^{n-k} = v' | L = k]$. Note that X^{2n} and the triple (Y^k, U^n, V^{n-k}) are in one-to-one correspondence of each other (to reconstruct X^{2n}, simply read the k distinct pairs from Y and fill them according to the locations ones in U and fill the remaining equal pairs from V). Finally, note that u, y, v and y', u', v' correspond to two input strings x and x' of identical Hamming weight ($w(x) = 2k + 2w(v)$) and hence of identical probability due to the exchangeability of X^{2n}. [Examples: $(y, u, v) = (01, 1100, 01) \Rightarrow x = (10010011)$, $(y, u, v) = (11, 1010, 10) \Rightarrow x' = (01110100)$.]

Computing the marginals, we conclude that both Y^k and U^n are uniform over their respective support set.[1] □

Lemma 28.3 (Ψ_t is an extractor)**.** *Let X^{2n} be exchangeable. Then $\Psi_t(X^{2n}) \overset{i.i.d.}{\sim} Bern(1/2)$ conditioned on $l(\Psi_t(X^{2n})) = m$.*

Proof. Note that $\Psi_t(X^{2n}) \in \{0,1\}^*$. It is equivalent to show that for all $s^m \in \{0,1\}^m$,

$$\mathbb{P}[\Psi_t(X^{2n}) = s^m] = 2^{-m}\mathbb{P}[l(\Psi_t(X^{2n})) = m].$$

Proceed by induction on t. The base case of $t = 1$ follows from Lemma 28.2 (the distribution of the Y part). Assume Ψ_{t-1} is an extractor. Recall that $\Psi_t(X^{2n}) = (\Psi_1(X^{2n}), \Psi_{t-1}(U^n), \Psi_{t-1}(V^{n-k}))$ and write the length as $L = L_1 + L_2 + L_3$, where $L_2 \perp\!\!\!\perp L_3 | L_1$ by Lemma 28.2. Then

$$\begin{aligned}
&\mathbb{P}[\Psi_t(X^{2n}) = s^m] \\
&= \sum_{k=0}^{m} \mathbb{P}[\Psi_t(X^{2n}) = s^m | L_1 = k]\mathbb{P}[L_1 = k] \\
&\overset{\text{Lemma 28.2}}{=} \sum_{k=0}^{m}\sum_{r=0}^{m-k} \mathbb{P}[L_1 = k]\mathbb{P}[Y^k = s^k | L_1 = k]\mathbb{P}[\Psi_{t-1}(U^n) = s_{k+1}^{k+r} | L_1 = k]\mathbb{P}[\Psi_{t-1}(V^{n-k}) = s_{k+r+1}^{m} | L_1 = k] \\
&\overset{\text{induction}}{=} \sum_{k=0}^{m}\sum_{r=0}^{m-k} \mathbb{P}[L_1 = k]2^{-k}2^{-r}\mathbb{P}[L_2 = r | L_1 = k]2^{-(m-k-r)}\mathbb{P}[L_3 = m-k-r | L_1 = k] \\
&= 2^{-m}\mathbb{P}[L = m].
\end{aligned}$$

□

[1] If X^{2n} is iid Bern(p), then V^{n-k} is iid Bern$(p^2/(p^2+q^2))$, since $L \sim$ Binom$(n, 2pq)$ and $\mathbb{P}[Y^k = y, U^n = u, V^{n-k} = v | L = k] = 2^{-k} \cdot \binom{n}{k}^{-1} \cdot (\frac{p^2}{p^2+q^2})^m (\frac{q^2}{p^2+q^2})^{n-k-m}$, where $m = w(v)$. In general we cannot say much more than the fact that V^{n-k} is exchangeable.

Next we compute the rate of Ψ_t. Let $X^{2n} \overset{\text{i.i.d.}}{\sim} \text{Bern}(p)$. Then by SLLN, $\frac{1}{2n} l(\Psi_1(X^{2n})) \triangleq \frac{L_n}{2n}$ converges a.s. to pq. Assume that $\frac{1}{2n} l(\Psi_{t-1}(X^{2n})) \xrightarrow{\text{a.s.}} r_{t-1}(p)$. Then

$$\frac{1}{2n} l(\Psi_{t-1}(X^{2n})) = \frac{L_n}{2n} + \frac{1}{2n} l(\Psi_{t-1}(U^n)) + \frac{1}{2n} l(\Psi_{t-1}(V^{n-L_n})).$$

Note that $U^n \overset{\text{i.i.d.}}{\sim} \text{Bern}(2pq)$, $V^{n-L_n} | L_n \overset{\text{i.i.d.}}{\sim} \text{Bern}(p^2/(p^2+q^2))$ and $L_n \xrightarrow{\text{a.s.}} \infty$. Then the induction hypothesis implies that $\frac{1}{2n} l(\Psi_{t-1}(U^n)) \xrightarrow{\text{a.s.}} r_{t-1}(2pq)$ and $\frac{1}{2(n-L_n)} l(\Psi_{t-1}(V^{n-L_n})) \xrightarrow{\text{a.s.}} r_{t-1}(p^2/(p^2+q^2))$. We obtain the recursion:

$$r_t(p) = pq + \frac{1}{2} r_{t-1}(2pq) + \frac{p^2+q^2}{2} r_{t-1}\left(\frac{p^2}{p^2+q^2}\right) \triangleq (T r_{t-1})(p), \tag{28.1}$$

where the operator T maps a continuous function on $[0,1]$ to another. Note that $f \le g$ pointwise then $Tf \le Tg$. Then it can be shown that r_t converges monotonically from below to the fixed-point of T, which turns out to be exactly the binary entropy function h. Instead of directly verifying $Th = h$, next we give a simple proof: Consider $X_1, X_2 \overset{\text{i.i.d.}}{\sim} \text{Bern}(p)$. Then $2h(p) = H(X_1, X_2) = H(X_1 \oplus X_2, X_1) = H(X_1 \oplus X_2) + H(X_1 | X_1 \oplus X_2) = h(p^2+q^2) + 2pq h(\frac{1}{2}) + (p^2+q^2) h(\frac{p^2}{p^2+q^2})$.

The convergence of r_t to h are shown in Fig. 28.1.

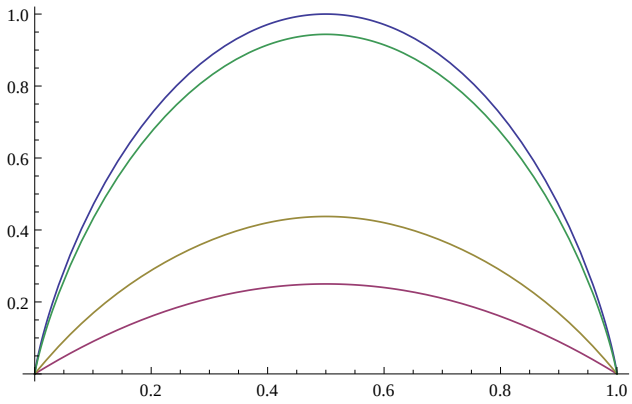


Figure 28.1: Rate function r_t for $t = 1, 4, 10$ versus the binary entropy function.

28.5 Bernoulli factory

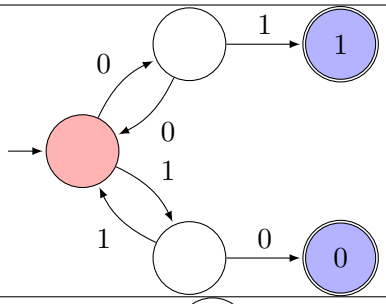
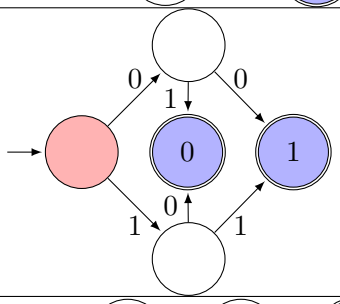
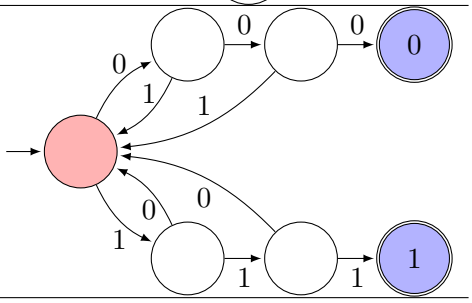
Given a stream of $\text{Bern}(p)$ bits with unknown p, for what kind of function $f : [0,1] \to [0,1]$ can we simulate iid bits from $\text{Bern}(f(p))$. Our discussion above deals with $f(p) \equiv \frac{1}{2}$. The most famous example is whether we can simulate $\text{Bern}(2p)$ from $\text{Bern}(p)$, i.e., $f(p) = 2p \wedge 1$. Keane and O'Brien [KO94] showed that all f that can be simulated are either constants or "polynomially bounded away from 0 or 1": for all $0 < p < 1$, $\min\{f(p), 1-f(p)\} \ge \{p, 1-p\}^n$ for some $n \ge 1$. In particular, doubling the bias is impossible.

The above result deals with what $f(p)$ can be simulated in principle. What type of computational devices are needed for such as task? Note that since $r_1(p)$ is quadratic in p, all rate functions r_t that arise from the iteration (28.1) are rational functions (ratios of polynomials), converging to the binary entropy function as Fig. 28.1 shows. It turns out that for any rational function f that satisfies $0 < f < 1$ on $(0,1)$, we can generate independent $\text{Bern}(f(p))$ from $\text{Bern}(p)$ using either of the following schemes [MP05]:

1. *Finite-state machine* (FSM): initial state (red), intermediate states (white) and final states (blue, output 0 or 1 then reset to initial state).

2. *Block simulation*: let A_0, A_1 be disjoint subsets of $\{0,1\}^k$. For each k-bit segment, output 0 if falling in A_0 or 1 if falling in A_1. If neither, discard and move to the next segment. The block size is at most the degree of the denominator polynomial of f.

The next table gives examples of these two realizations:

Goal	Block simulation	FSM
$f(p) = 1/2$	$A_0 = 10; A_1 = 01$	
$f(p) = 2pq$	$A_0 = 00, 11; A_1 = 01, 10$	
$f(p) = \frac{p^3}{p^3+q^3}$	$A_0 = 000; A_1 = 111$	

Exercise: How to generate $f(p) = 1/3$?

It turns out that the only type of f that can be simulated using either FSM or block simulation is rational function. For $f(p) = \sqrt{p}$, which satisfies Keane-O'Brien's characterization, it cannot be simulated by FSM or block simulation, but it can be simulated by pushdown automata (PDA), which are FSM operating with a stack [MP05].

What is the optimal Bernoulli factory with the best rate is unclear. Clearly, a converse is the entropy bound $\frac{h(p)}{h(f(p))}$, which can be trivial (bigger than one).

28.6 Related problems

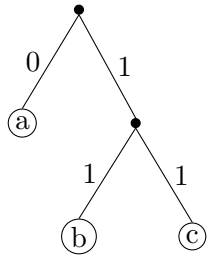
28.6.1 Generate samples from a given distribution

The problem of how to turn pure bits into samples of a given distribution P is in a way the opposite direction of what we have been considering so far. This can be done via Knuth-Yao's tree algorithm:

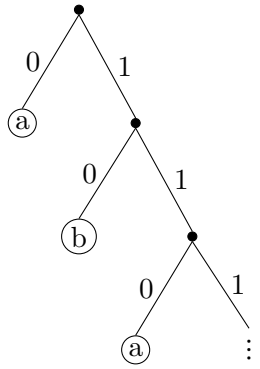
Starting at the root, flip a fair coin for each edge and move down the tree until reaching a leaf node and outputting the symbol. Let L denote the number of flips, which is a random variable. Then $H(P) \le \mathbb{E}[L] \le H(P) + 2\texttt{bits}$.

Examples:

- To generate $P = [1/2, 1/4, 1/4]$ on $\{a, b, c\}$, use the finite tree: $\mathbb{E}[L] = 1.5$.



- To generate $P = [1/3, 2/3]$ on $\{a, b\}$ (note that $2/3 = 0.1010\ldots, 1/3 = 0.0101\ldots$), use the infinite tree: $\mathbb{E}[L] = 2$ (geometric distribution)



28.6.2 Approximate random number generator

The goal is to design $f : \mathcal{X}^n \to \{0,1\}^k$ s.t. $f(X^n)$ is *close to* fair coin flips in distribution in certain distances (TV or KL). One formulation is that $D(P_{f(X^n)} \| \text{Uniform}) = o(k)$.

Intuitions: The connection to lossless data compression is as follows: A good compressor squeezes out all the redundancy of the source. Therefore its output should be close to pure bits, otherwise we can compress it furthermore. So good lossless compressors should act like good approximate random number generators.

6.441 Information Theory

Spring 2016