Entropy and Guesswork

David Malone (CNRI DIT)

16 October 2002

Measuring Randomness

Start with a simple measurement of uncertainty.

You have a program with n bits of random initial state then the program can have at most 2^n different ways it can run.

Calculate minimum randomness for:

Lotto Quick Pick:

$$\frac{42!}{6!36!} = 5245786$$

Requires $\lg(5245786) \approx 23$ bits.

Shuffling Cards:

$$52! = 8.0658\ldots \times 10^{67}$$

Requires $\lg(52!) \approx 226$ bits (29 bytes).

Shuffling Election votes: For n votes we need about this many bits:

$$\begin{aligned}\lg(n!) &= \sum_{m=1}^{n} \lg(m) \\ &\approx \int_{1}^{n} \lg(m)\, dm \\ &\approx n \lg(n)\end{aligned}$$

For 1,000,000 votes that is about 20 million bits — or about 1.25MB!

Entropy Measuring Uncertainty

A source which produces symbols $a \in \mathbb{A}$ with probability p_a has entropy

$$h(p) = \sum_{a \in \mathbb{A}} p_a \lg \frac{1}{p_a}.$$

Shannon proved $h(p)$ is the average number of bits required to encode a message from that source. It adds for independent sources.

Entropy is often interpreted as the amount of information or uncertainty associated with a source.

Guessing and Cryptography

Encryption requires selecting an algorithm and a key. Great care is invested in designing algorithms and so it may be easier to attack the key.

- A *brute force attack* involves trying every key one after another. Your key space must be big to make this impractical.
- A *dictionary attack* uses the fact that people are more likely to choose real words as keys.

Pseudo-random numbers used by computers can be subject to dictionary-like attacks if seeded badly.

Entropy and Guessing

Entropy is a measure of uncertainty. Does it capture how hard it is to guess a number? From the `sci.crypt` FAQ:

> We can measure how bad a key distribution is by calculating its entropy. This number E is the number of "real bits of information" of the key: a cryptanalyst will typically happen across the key within 2^E guesses. E is defined as the sum of $-p_K \log_2 p_K$, where p_K is the probability of key K.

Can we check this?

The quickest way to guess a symbol is to first guess the most likely value, and then proceed towards the least likely value. Label p_k in decreasing order with integers then the expected amount of guessing time or *guess work* is

$$G(p) = \sum_k p_k k.$$

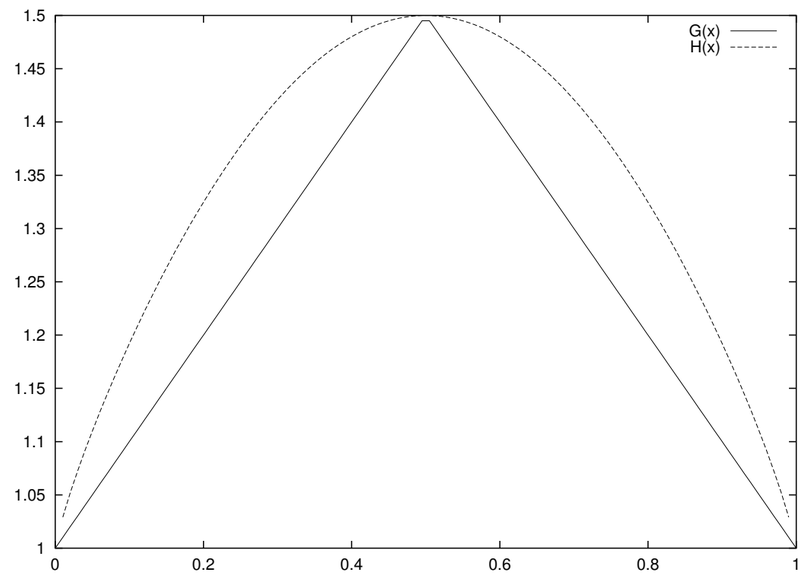
We want to compare this to an entropy based estimate,

$$H(p) = \frac{2^{h(p)} + 1}{2},$$

because guessing from r equally likely options takes $(r+1)/2$ guesses.

Bernoulli Source

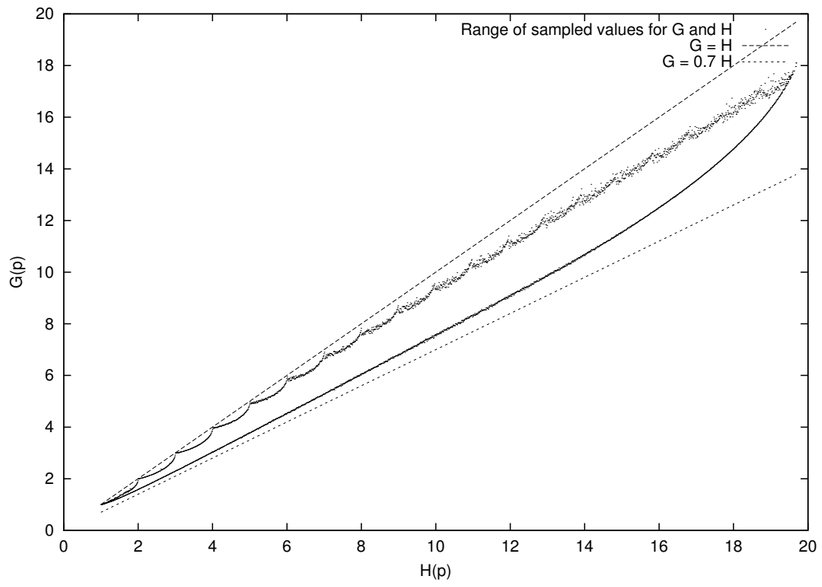
Here $\mathbb{A} = \{0, 1\}$ and
$\mathbb{P}(0) = p, \mathbb{P}(1) = q = 1 - p.$



$$G(p) = \begin{cases} 1p + 2(1-p) & p \geq 0.5 \\ 1(1-p) + 2p & p < 0.5 \end{cases}.$$

$$H(p) = 2^{-p \lg p - (1-p) \lg (1-p)} = p^{-p} q^{-q}.$$

Simulation



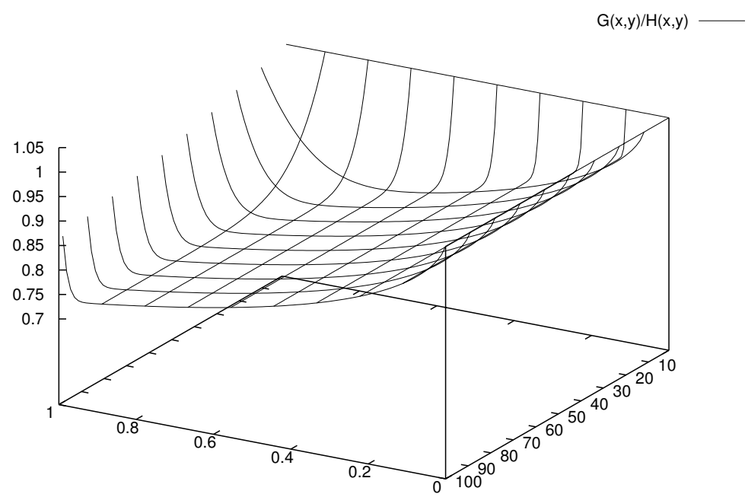
Simulated by choosing up a random distribution on up to 20 symbols.
Hypothesis:

$$0.7H(p) \leq G(p) \leq H(p).$$

Show $0.7H(p) \leq G(p)$ with Lagrange Multipliers: Fix $G(p)$ and find extrema of $H(p)$ at

$$p_k = C\lambda^k,$$

(luckily, a decreasing sequence).



Then evaluate G and H explicitly.

$$\lim_{n\to\infty,\lambda\to 1} \frac{G}{H} = \lim_{\lambda\to 1} \frac{2}{1-\lambda+\lambda^{\lambda/(\lambda-1)}} \to \frac{2}{e}$$

Massey also shows that the upper bound $G(p) \leq H(p)$ isn't, using the sequence

$$p_k = \begin{cases} 1 - \dfrac{\beta}{n} & k = 1 \\ \\ \dfrac{\beta}{n^2 - n} & 2 \leq k \leq n \end{cases},$$

and letting n become large. This sequence has an entropy tending to zero, but a constant guess work.

So, entropy is a lower bound on guess work, but not an upper bound. Lucky for those cryptologists...

How did this incorrect idea get into the folklore?

Asymptotic Equipartition

One important place where entropy arises is in the Asymptotic Equipartition Property (AEP). If we have n independent identical sources and we look at their combined output in $\mathbb{A}^n$ then the set

$$T_\epsilon^{(n)} = \left\{ a \in \mathbb{A}^n : |\mathbb{P}(a) - 2^{-nh(p)}| < \epsilon \right\}$$

has the following properties:

- $\mathbb{P}(T_\epsilon^{(n)}) \to 1$ as $n \to \infty$.
- $|T_\epsilon^{(n)}| \approx 2^{nh(p)}$.

These elements are considered 'typical'.

AEP and Guessing

Plaim suggests that the link between guesswork and entropy may have arisen via the AEP. Remember, the AEP says that we can find a set of words $T_\epsilon^{(n)}$ so that the probability of each word is about $2^{-nh(p)}$ and by making n big enough we can make $\mathbb{P}(T_\epsilon^{(n)})$ close to 1. Ignoring the atypical words,

$$G(p) = \sum_k p_k k = \sum_{T_\epsilon^{(n)}} 2^{-nh(p)} k = \frac{2^{nh(p)} + 1}{2}.$$

Setting $n = 1$ then produces folklore...

Can we salvage a result if n large?

Look at sets of symbols $(a_1, \ldots a_n)$ in $\mathbb{A}^n$ with probability $p_{a_1} \ldots p_{a_n}$. Guess in the same way as before and only stop if all symbols correct.

To evaluate $G_n(p)$ calculate all the products $p_{a_1} \ldots p_{a_n}$ and sort them, then

$$G_n(p) = \sum_k p_{a_{k,1}} \ldots p_{a_{k,n}} k.$$

Evaluating $H_n(p)$ is much easier 'cos the entropy of independent sources adds:

$$H_n(p) = \frac{2^{h_n(p)} + 1}{2} = \frac{2^{nh(p)} + 1}{2} = \frac{H(p)^n + 1}{2}.$$

Is $G_n(p) \approx H_n(p)$?

Product Bernoulli Source

Most cases are hard: have to sort product of probabilities. In Bernoulli case, if $0 \leq p \leq 0.5$, we know $p^k q^{n-k}$ is in non-increasing order. Thus,

$$G_n(p) = \sum_{k=0}^{n} f(k,n) p^k q^{n-k} \binom{n}{k}$$

where

$$f(k,n) = \sum_{j=0}^{k-1} \binom{n}{j} + \frac{1}{2}\binom{n}{k}.$$

$H_n(p)$ grows exponentially so consider

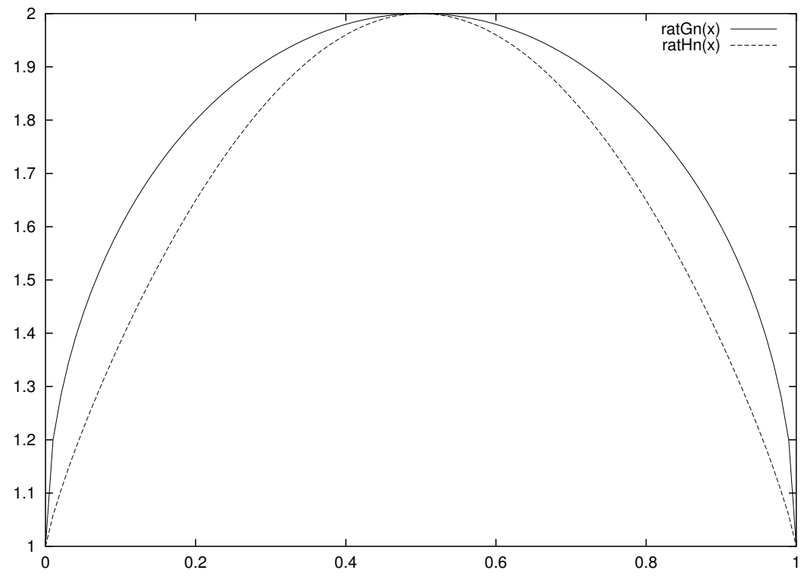
$$\lim_{n\to\infty} \frac{1}{n} \log G_n(p).$$

We find that

$$G_n(p) \asymp \left(\left(\sqrt{p} + \sqrt{q} \right)^2 \right)^n$$

and know that

$$H_n(p) \asymp \left(p^{-p} q^{-q} \right)^n .$$



This generalises as you would hope:

$$G_n(p) \asymp \left((\sqrt{p_1} + \sqrt{p_2} + \ldots)^2 \right)^n$$

Amazingly, $\lg \left(\left(\sqrt{p_1} + \sqrt{p_2} + \ldots \right)^2 \right)$ is a generalisation of Shannon entropy called Rényi entropy.

Has also been generalised to Markov Chains (WGS) and to more general spaces (WGS+CP).

Collecting Randomness

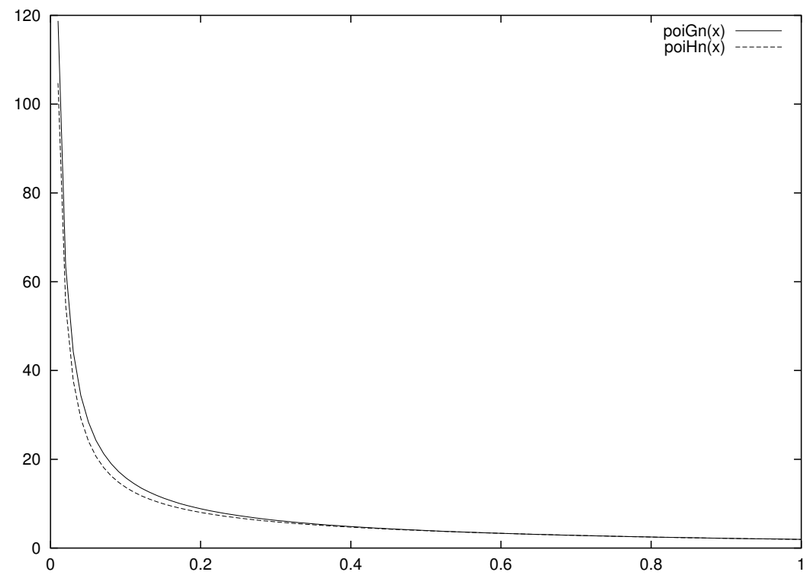
Various techniques:

- Timing radioactive decays.
- Radio Noise (`www.random.org`).
- Timing interrupts.
- Compressed machine state (`egd.sourceforge.net`).
- Intel 8XX chipset.

Offered on Unix via `/dev/random`.
Managed via techniques such as Yarrow.

Application

Collecting randomness by measuring background radiation. Watch for time interval T, no decays $a = 0$ otherwise $a = 1$. Poisson distributed so $p = e^{-T}$. Do optimal T for long term rate of entropy and guess work collection differ?



Another Guessing Problem

Help desk answers m types of question (frequency $q_1, \ldots q_m$).

'Phoning to random person dealing with 1 type, transfered until right type.

If trained in proportion $p_1, \ldots p_m$, how many transfers?

$$\begin{aligned} E[\text{search time}] &= \sum_{k=1}^{m} q_k \sum_{l=1}^{\infty} p_k (1-p_k)^l l \\ &= \sum_{k=1}^{m} \frac{q_k}{p_k}. \end{aligned}$$

How should we train the helpdesk?

Optimising p_k gives

$$\left(\frac{p_k}{p_l}\right)^2 = \left(\frac{q_k}{q_m}\right),$$

so $p_k \propto \sqrt{q_k}$ and

$$(\sum_k \sqrt{q_k})^2.$$

This is also a modle for indexing in P2P networks.

Moral

1. Don't always believe simulations.
2. Randomness is important: but be careful to define randomness.
3. The crypto guys goofed: Entropy is easier than guess work.
4. Mathematical abstractions usually find an application.

Furtue

1. Calculate G, H for people.
2. How does guesswork combine?
3. Link between Rényi and guessing?