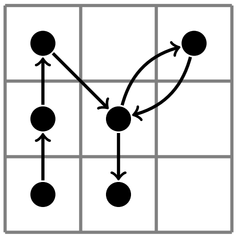
Nondeterministic Entropy

Mathias Winther Madsen

January 13, 2015

Problem 1. A swipe code is a pattern drawn on a 3×3 chess board by dragging your finger across the board from field to field.



How many swipe codes of length $t = 4$ are there? Approximately how many swipe codes of length $t = 1000$ are there?

1 Nondeterministic Processes

Definition 2. A **nondeterministic process** is a collection of infinite sequences from some alphabet $\mathbb{A}$. We call the individual members of the nondeterministic process **sample paths**.

We will mostly focus on nondeterministic processes over the binary alphabet $\mathbb{A} = \{0, 1\}$. Some examples of such processes are:

Example 3. The unconstrained binary process: The set of all binary sequences, without any restrictions, such as

$$0\,0\,0\,1\,0\,0\,1\,0\,1\,0\,0\,0\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,\ldots$$

Example 4. The Morse code process: The binary sequences without any consecutive 1s, such as

$$1\,00\,1\,000\,1\,0000\,1\,0\,1\,000000\,1\,0\,1\,000\ldots$$

Example 5. The fencepost process: The binary sequences with equally-spaced 1s, such as

$$000\,1\,000\,1\,000\,1\,000\,1\,000\,1\,000\,1\,000\ldots$$

Example 6. The outspender process: The binary sequences that alternate between growing strings of 0s and 1s, such as

$$0\,111\,00000\,111111\,0000000000\ldots$$

A nondeterministic process over a finite alphabet can be visualized as a tree. For instance, a part of the tree corresponding to the Morse code process is shown in Figure 1). Each infinite branch though this tree corresponds to one particular element of this process.

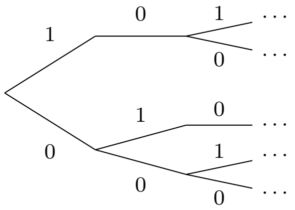


Figure 1: At depth $t = 3$, the Morse code process has a width of $N(3) = 5$.

Definition 7. We will say that a nondeterministic process **admits** a certain finite string if that finite string is an initial segment of a sample path from the process. If it does, we will call the string a **sample** from the process. The number of samples of length t is denoted $N(t)$, and we call N the **growth function** for the process.

For the unconstrained binary process, the growth function is the exponential function $N(t) = 2^t$, since all 2^t binary strings of length t are admitted by that process. Increasing the sample size by 1 thus doubles the number of samples consistent with the unconstrained process. As we shall see, such exponential growth is characteristic of many processes of practical interest.

For the Morse code process, $N(t) < 2^t$ for $t \geq 2$, since not all strings are admitted. For instance, the sample 11 is not admitted because it contains two consecutive 1s, so $N(2) = 3$ rather than 4. A few more values of this growth function are:

t	1	2	3	4	5	6	7	$\cdots$
$N(t)$	2	3	5	8	13	21	34	$\cdots$

These are in fact the Fibonnaci numbers, an exponentially increasing series which roughly grows by a factor of about 1.618 every time we increment t by one. We can thus approximate N by $N(t) \approx N(1) \times 1.618^t$, or

$$N(t) \;\approx\; N(t) \times 2^{0.694t}$$

since $2^{0.694} \approx 1.618$, or $\log_2 1.618 \approx 0.964$.

The Morse code process thus admits roughly as many samples for $t = t_0$ as the unconstrained process admits for $t = 0.694\, t_0$. For purposes of data compression, this means that we can use the samples of length 694 from the unconstrained process as "names" for the samples of length 1000 from the Morse code process. We can, in other words, save about $1000 - 694 = 206$ bits of storage space by encoding the Morse code samples as unconstrained binary strings.

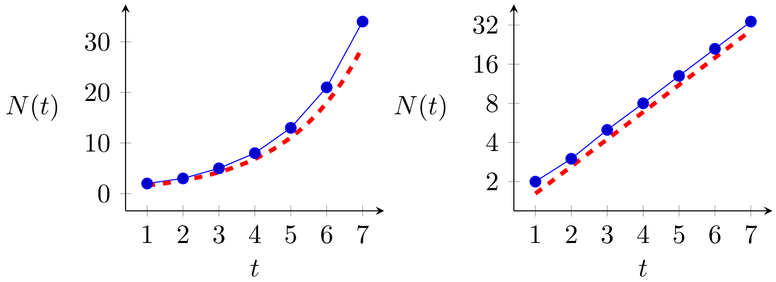


Figure 2: The growth function $N(t)$ of the Morse code process (connected dots) along with its exponential regression line $2^{0.694t}$ (dashed).

2 The Entropy Rate

These observations about asymptotic exponential growth give rise to the following definition, first proposed by Shannon (1948):

Definition 8. The **entropy rate** of a nondeterminstic process is

$$H \;=\; \lim_{t\to\infty} \frac{\log N(t)}{t}$$

if the limit exists. When log designates the binary logarithm $\log_2$, the entropy rate is measured in **bits**; if it designates the natural logarithm ln, in **nats**.

The entropy rate can be seen as a measure of how nondeterministic or "choicy" a process is, relative to the process that admits two letters per unit of time. For a nondeterministic process over a finite alphabet $\mathbb{A}$, the entropy rate can at most be $\log |\mathbb{A}|$, the rate of an unconstrained process over $\mathbb{A}$.

Here are some examples:

Example 9. The unconstrained binary process has an entropy rate of $H = 1$.

Example 10. The Morse code process has an entropy rate of approximately

$$H \;=\; \lim_{t\to\infty} \frac{\log N(1) + \log 2^{0.694t}}{t} \;=\; 0.694.$$

Note that the initial condition $N(1)$ actually makes no difference for the entropy rate. This reflects the fact that the slope of a line through $(1, \log N(1))$ and $(t, \log N(t))$ is almost independent of $N(1)$ once t gets large enough.

Example 11. The fencepost process has an entropy rate of $H = 0$. This is because a sample from the fencepost process is completely defined by the position of the first 1 in the sample. Once that position has been revealed, the rest of the sample is contains no further information.

In a sample of length t, there are t different places one can place this first fencepost, in addition to the option of not having any 1s in the sample at all. The fencepost process thus admits $N(t) = t + 1$ different samples of length t, and its entropy rate is

$$H \;=\; \lim_{t\to\infty} \frac{\log(t+1)}{t} \;=\; 0.$$

In the limit, this process is thus as predictable as a deterministic process.

Example 12. The outspender process also has an entropy rate of $H = 0$, but seeing this is a little more involved than the previous examples.

A sample from the outspender process contains a sequence of runs, alternating between 0s and 1s. Since these runs are required to increase in size after each alternation, a sequence of r runs has to consume at least

$$0 + 1 + 2 + 3 + \cdots + r \;=\; \frac{r(r+1)}{2} \;\geq\; \frac{1}{2}r^2$$

sample elements. A sample of length $t \geq r^2/2$ can therefore accommodate at most $r \leq \sqrt{2t}$ alternations. There are, in other words, $\sqrt{2t} + 1$ possibilities for how many completed runs such a sample may contain (including the option 0).

Once the number of alternations has been fixed, they also have to be distributed into the sample. Since there are $t+1$ spaces between t sample elements, the number of ways we can choose these positions is at most the binomial coefficient $C(t+1, \sqrt{2t})$, where the square root is rounded up to the nearest integer. (It is in fact somewhat lower due to the restrictions on the run lengths, but this approximation will be sufficiently low for our purposes.)

The number of samples from the outspender process is thus bounded by

$$N(t) \;\leq\; \left(\sqrt{2t}+1\right) \times \binom{t+1}{\sqrt{2t}},$$

and thus

$$\frac{\log N(t)}{t} \;\leq\; \frac{1}{t}\log\left(\sqrt{2t}+1\right) + \frac{1}{t}\log\binom{t+1}{\sqrt{2t}}.$$

The first of these terms tends to 0 as $t \to \infty$. The second term can be estimated by means of Stirling's approximation, $\ln n! \approx n \ln n - n$. After using this approximation and doing some algebra, we can show that the only significant terms are

$$\frac{1}{t}\log\binom{t+1}{\sqrt{2t}} \;\approx\; \log(t+1) - \log(t+1-\sqrt{2t}) \;=\; \log\frac{t+1}{t+1-\sqrt{2t}}.$$

This expression tends to 0 as $t \to \infty$. Hence, $\log N(t)/t$ tends to $0+0=0$, and the entropy rate of the outspender process is $H = 0$.

3 Processes Without Entropy Rates

One way of thinking of the entropy rate is as a geometric average. If a population has increased from a size of 1 to a size of $N(t)$ over a period of t units of time, and if we believe that it grew by roughly the same multiplicative factor in each unit of time, then that factor must have been $N(t)^{1/t} = 2^H$. This geometric average contrasts with the arithmetic average $N(t)/t$ which characterizes the growth of a population that grows by constant additive increments rather than constant multiplicative factors.

This also means that H is the rate we need to assume in order to fit the exponential growth 2^{Ht} to the function $N(t)$. Graphically, this corresponds to fitting a straight line to the points $(t, N(t))$ and then reading off its slope.

Tt is important to remember that a nondeterministic process can fail to have an entropy rate. This can happen for two reasons:

1. If Ω is infinite, $N(t)$ can grow superexponentially. For instance, let $\Omega = \mathbb{N}$ and consider the process consisting of all sequences whose tth entry is an element of the set $\{1, 2, 3, \ldots, t\}$. An example of sample path from this process is

   $$1\ 2\ 3\ 2\ 5\ 5\ 2\ 6\ 3\ 10\ 10\ 3\ 7\ 13\ 7\ 5\ 10\ 14\ 18\ 20\ \ldots$$

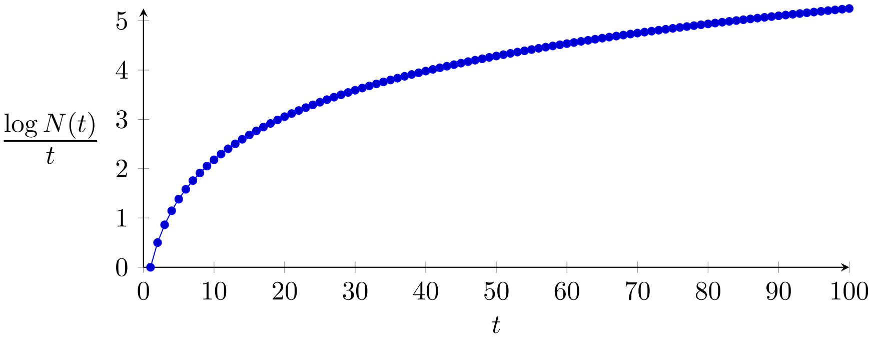


Figure 3: An infinite alphabet can cause the average uncertainty to diverge.

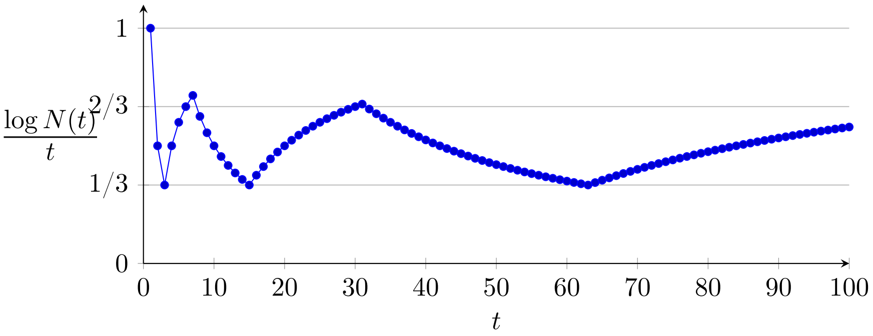


Figure 4: Large oscillations in levels of predictability can mean that no exponential tendency appears in the number of possible sequences.

For this process, increasing t by one corresponds to multiplying $N(t)$ by $t+1$, so we cannot approximate the growth of this process by any exponential function with a fixed rate H. In fact, $N(t) = t!$, and $\log N(t)$ thus grows as $t \log t$ in the limit, rather than a linear function tH, as an exponential approximation requires.

2. Convergence can also fail because $\log N(t)/t$ oscillates up and down without approaching a limit. For instance, consider a nondeterministic process admitting first 1 unconstrained bit, then 2 deterministic bits, then 4 unconstrained bits, then 8 deterministic bits, and so on. A sample from this process is

$$1\,\underline{00}\,1011\,\underline{00000000}\,0101100101110100\,\underline{0000000000000000}\ \ldots$$

where the deterministic bits are underlined for clarity. In this process, the runs of deterministic bits are always so long that they swamp everything that came before, bringing $\log N(t)/t$ down to roughly 1/3; but the subsequent runs of nondeterministic bits are also longer than anything that came before and bring it back up to roughly 2/3 again (cf. Fig. XXX). The convergence of $\log N(t)/t$ is thus spoiled by the superlinear oscillations of $\log N(t)$.

4 Growth Functions and Linear Algebra

In many important cases, we can find the entropy rate of a process by doing a bit of linear algebra. In particular, this is the case for **Markov processes**, that is, processes whose restrictions are defined in terms of which symbols can and cannot occur next to each other.

Consider for instance the process over the alphabet $\mathbb{A} = \{0, 1, 2\}$ defined by the transition graph in Figure XX. This process requires that the symbols 0 and 1 never occur twice in a row, and that the symbol 2 is never followed by the symbol 0. We could also represent this transition graph as the matrix

$$T \;=\; \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix},$$

where $T_{ij} = 1$ means that the transition from j to i is allowed.

We can also use this matrix to count the number of admissible sequences of a given length. We just need to specify, in the form of a vector, which samples of length $t = 1$ the process admits, and then proceed by matrix multiplication. For instance, suppose we require that the sample paths for this process always start with one of the symbols 0 or 2; we can then count the sequences of length 2, 3, and 4 by matrix multiplication:

$$T\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \;=\; \begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix}; \quad T^2\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 4 \end{pmatrix}; \quad T^3\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 6 \\ 8 \end{pmatrix}.$$

These product tell us how many sequences there are, split up according to the last symbol in the sequence. For instance, for $t = 3 + 1$, there are 2 samples terminating with the symbol 0, 6 with the symbol 1, and 8 with the symbol 2. There are thus a total of $N(4) = 2 + 6 + 8 = 16$ sequences of length 4 that start with either 0 or 2.

The formulation of the problem in terms of matrix multiplication also gives us access to some tools from linear algebra. In particular, we know that many linear mappings have **eigenvectors**, that is, vectors which are only stretched by a constant λ, the **eigenvalue**, when the mapping is applied:

$$Tv \;=\; \lambda v.$$

It is useful to imagine the initial condition s as decomposed into a linear combination of eigenvectors, since the effect of the linear mapping will then be to simply increase each component by the relevant eigenvalue:

$$\begin{aligned} Ts &= T(a_1v_1 + a_2v_2 + \cdots + a_nv_n) \\ &= a_1Tv_1 + a_2Tv_2 + \cdots + a_nTv_n \\ &= \lambda_1a_1v_1 + \lambda_2a_2v_2 + \cdots + \lambda_na_nv_n. \end{aligned}$$

Repeated application of the linear mapping will then correspond to repeated multiplication by the eigenvalues. All the components of the initial condition will thus change exponentially fast:

$$T^ts \;=\; \left(\lambda_1^t\right)a_1v_1 \;+\; \left(\lambda_2^t\right)a_2v_2 \;+\; \cdots \;+\; \left(\lambda_n^t\right)a_nv_n.$$

However, since the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ can have different sizes, the components may not grow at the same rate. In particular, if one of the eigenvalues is larger than the other ones, the corresponding component outgrow the others at an exponential rate. In the limit, the effect of applying the mapping one more time will thus be a stretching of the input vector in the direction of the dominant eigenvector, plus some other effects that are negligible by comparison.

As discussed in many textbooks on linear algebra, we can find the dominant eigenvector v^* by selecting the largest root λ^* of the equation $\det(T - \lambda I) = 0$ and then solving the equation $Tv = \lambda^*v$ for v. If this is impractical, we can also simply approximate v^* by applying T lots of times to an arbitrary initial condition.

For the matrix T above, $\lambda^* = 2$, and the corresponding eigenvector is

$$v^* \;=\; \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

or any vector parallel to it. For very large t, this process will thus admit vectors terminating in 0, 1, and 2 in the proportions $1:2:3$, and $N(t)$ will grow by a factor of about $\lambda^* = 2$ every time we increment t by one. Thus, $H = \lambda^* = 2$.