Measures of Complexity

• Entropy
• AIC (Kolmogorov Complexity)
• EFFECTIVE COMPLEXITY
• Total Information
• Logical Depth
• Statistical Complexity
• Size
• Fractal Dimension
Gell-Mann’s Effective Complexity

- The length of the shortest description of a set’s regularities
- \( EC(x) = K(r) \) where \( r \) is the set of regularities in \( x \) and Kolmogorov Complexity (or AIC), \( K(r) \), is the length of a concise description of a set
- Highest for entities that are not strictly regular or random
Algorithmic Complexity (AIC)
(also known as Kolmogorov-Chaitin complexity)

• **Kolmogorov complexity** or Algorithmic Information Content (AIC), written $K(x)$, is the length, in bits, of the smallest program that when run on a Universal Turing Machine outputs (prints) $x$ and then halts.

• Example: What is $K(x)$
  – where $x$ is the first 10 even natural numbers?
  – where $x$ is the first 5 million even natural numbers?

• Possible representations where $n$ is the length of $x$
  – $0, 2, 4, 6, 8, 10, 12, 14, 16, 18, \ldots (2n - 2)$
  – for (j = 0; j < n: j++) printf("%d\n", j * 2)
Algorithmic Complexity (AIC)  
(also known as Kolmogorov-Chaitin complexity)

• *Kolomogorov* complexity or Algorithmic Information Content (AIC), written $K(x)$, is the length, in bits, of the smallest program that when run on a Universal Turing Machine outputs (prints) $x$ and then halts.

• Example: What is $K(x)$
  – where $x$ is the first 10 even natural numbers?
  – where $x$ is the first 5 million even natural numbers?

• Possible representations where $n$ is the length of $x$
  – 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, … $(2n - 2)$  \[K(x) = O(n \log n) \text{ bits}\]
  – for (j = 0; j < n: j++) printf(“%d
”, j * 2)  \[K(x) = O(\log n) \text{ bits} \]

*Complexity of program size (length), not program running time*
Algorithmic Complexity (AIC)
(also known as Kolmogorov-Chaitin complexity)

• AIC formalizes what it means for a set of numbers to be compressible
  – Data that are redundant can be compressed and have lower AIC.
  – Random strings are incompressible, therefore contain no regularities to compress
    • \( K(x) = |\text{Print}(x)| \)

• Implication: The more random a system, the greater its AIC (and greater entropy)

• Contrast with Statistical simplicity
  Random strings are simple because you can approximate them statistically
  – Coin toss, random walks, Gaussian (normal distributions)
  – You can compress random numbers with statistical descriptions and only a few parameters
• $s_1 = \underbrace{1111111111111111}_{\text{16 times}}$
  
  for $i:=1:16$
  print ‘1’

  $K(s) \in O(\log n)$ where $n = \text{length}(s_1)$
  The dominant term is the number “16” whose representation in bits will require $\log_2 16 = 4$ bits

• $S_2$ is a string of 1’s, $\text{length}(s_2) = 1$ billion
  
  for $i:=1:1,000,000,000$
  print ‘1’

  $K(s_1)$ is $O(\log n)$ where $n = \text{length}(s_2)$
  $K(s_2)$ is $O(\log_2 10^9)\quad(\text{approximately 30 bits})$

Constant terms such as the representation of the print statement are ignored

The minimum $K(x)$, given $\text{length}(x) = n$, is $\log(n)$.

$s_3 = 0101010101010101$
$s_4 = 011011011011011$
$s_5 = 000100001000100001$
$s_6 = 01110111011000110110101$
$s_3 = 0101010101010101$
for $i := 1:8$
  print '01'
2 bits are needed to represent the pattern, 3 bits to represent the 8 repetitions
$K(s_3) = O(\log(n))$

$s_4 = 0110110110110111$
for $i := 1:5$
  print '011'
3 bits to represent the repeating pattern, $\log(5)$ bits to represent the number of repetitions
$K(s_4) = O(\log(n))$

$s_5 = 000100001000100001$
for $c = 1:2$
  print '00010001'
8 bits to represent the repeating pattern, 1 bit to represent the repetitions. Here the length of the pattern is $\frac{1}{2}n$, so
$K(s_5) = O(n)$

$s_6 = 01110111011000110110101$
There are two problems with AIC

– Calculation of $K(x)$ depends on the machine we have available (e.g., what if we have a machine with an instruction “print the first 10 even natural numbers”?)
  • COMPLEXITY DEPENDS ON CONTEXT

– Determining $K(x)$ for arbitrary $x$ is uncomputable
Problem 1: $K(x)$ depends on the programming language
Resolution: optimal specification functions can be defined so that
“The complexity of an object $x$ is invariant (up to an additive constant independent of $x$) under transition from one optimal specification function to another.”
(Li & Vitanyi “An Introduction Kolmogorov Complexity and it’s Applications” 2008)

http://jeremykun.com/2012/04/21/kolmogorov-complexity-a-primer/

Lemma: For any strings $w, x$ and any language $L$, we have $K_L(w | x) \leq |w| + c$ for some constant $c$ independent of $w, x$, and $K_L(w) \leq |w| + c'$ for some constant $c'$ independent of $w$.

Proof. The program which trivially outputs the desired string has length $|w| + c$, for whatever constant number of letters $c$ is required to dictate that a string be given as output. This is clearly independent of the string and any input. □

It is not hard to see that this definition is invariant under a choice of language $L$ up to a constant factor. In particular, let $w$ be a string and fix two languages $L, L'$. As long as both languages are universal, in the sense that they can simulate a universal Turing machine, we can relate the Kolmogorov complexity of $w$ with respect to both languages. Specifically, one can write an interpreter for $L$ in the language of $L'$ and vice versa. We saw a
Problem 2: Identifying the shortest program to print the string is uncomputable

Resolution: None

It is not possible to determine the amount of randomness in any arbitrary string
Regularities can be difficult to identify

- $s_6 = 01110111011000110110101$
- It looks like a random string, so $K(s_6) = \text{length}(s_6)$
  
  But it's not!

- What is the shortest program that would produce this string?
A short computer program (of length \( L_{fb} \)) that produces the Fibonacci series can generate \( s_6 \), so \( K(s_6) \) can be reduced to \( \max \left( L_{fb}, \log(n) \right) \) where \( n \) specifies the length of the Fibonacci series to be printed.
What About

$s7=00100100001111110110101010001000100001011010001$
$100001000110100...$?

This is a statistically random string.
It is also the binary representation of the first decimals of pi.

A short program could generate this random string.
Gell-Mann:
In Evolution “Frozen Accidents” cause regularities

Identify “frozen accidents” (regularities) in genomes
Calculate Effective Complexity as the AIC of the regularities
Gell-Mann’s Effective Complexity

- The length of the shortest description of a set’s regularities
- \( EC(x) = K(r) \) where \( r \) is the set of regularities in \( x \) and Kolmogorov Complexity (or AIC), \( K(r) \), is the length of a concise description of a set
- Highest for entities that are not strictly regular or random

![Algorithmic Complexity & Entropy vs Randomness](image)

![Effective Complexity vs Randomness](image)
What’s a regularity?

- Gell Mann suggests one formal way to identify regularities
- Determine mutual AIC between parts of the string
  - If \( x = [x_1, x_2] \)
  - \( K(x_1, x_2) = K(x_1) + K(x_2) - K(x) \)
  - The sum of the AICs of the parts – the AIC of the whole
  - Eg. 10010 10011 10010 - the whole has more regularity than the sum of the regularities in the parts
  - Regularities exist when \( K(x_1, x_2) > 0 \)

- Identify the regularities, \( r \),
- Calculate \( K(r) \), the AIC of the regularities
- Effective complexity = \( K(r) \)

Here \( r = 10010 \)
AIC of 10010 which is \( O(5) \)
Effective Complexity is \( O(5) \)
Logical Depth

• Bennett 1986;1990:
  – The *Logical depth* of *x* is the run time of the shortest program that will cause a UTM to produce *x* and then halt.
  – Logical depth is not a measure of randomness; it is small both for trivially ordered and random strings.

• Drawbacks:
  – Uncomputable.
  – Loses the ability to distinguish between systems that can be described by computational models less powerful than Turing Machines (e.g., finite-state machines).

• Ay et al 2008, recently proposed proof that strings with high effective complexity also have high logical depth, and low effective complexity have small logical depth.
Total Information

- Alternative approach in Gell-Mann & Lloyd 1998
- \( EC(x) = K(E) \) where \( E \) is the set of entities of which \( x \) is a typical member
- Then \( K(x) \) is the length of the shortest program required to specify the members of a the set of which \( x \) is a typical member
- Effective complexity measures knowledge—the extent to which the entity is nonrandom and predictable
- Total Information is Effective complexity, \( K(E) \), + the Shannon Information of the peculiarities (remaining randomness)
- \( TI(x) = K(E) + H(x) \)
- There is a tradeoff between the effective complexity (the completeness of a description of the regularities) and the remaining randomness
- Ex: 10010 10011 10010 10011 10010 10011
Summary of Complexity Measures

• **Information-theoretic methods:**
  – Shannon Entropy
  – Algorithmic complexity
  – Mutual information

• **Effective Complexity:**
  – Neither regular nor random entities have high Effective Complexity

• **Total Information: Effective Complexity + Entropy**
  • AIC of regularities + entropy of what remains

• **Computational complexity:**
  – How many resources does it take to compute a function?

• **The language/machine hierarchy:**
  – How complex a machine is needed to compute a function?

• **Logical depth:**
  – Run-time of the shortest program that generates the phenomena and halts.

• **Asymptotic behavior of dynamical systems:**
  – Fixed points, limit cycles, chaos
Defining Complexity
Suggested References

- Ay, Muller & Szkola, Effective Complexity and its Relation to Logical Depth, ArXiv (2008)
Kolmogorov Complexity References

• Kolmogorov Complexity With Python
  https://www.youtube.com/watch?v=KyB13PD-UME
• http://en.wikipedia.org/wiki/Kolmogorov_complexity
• http://www.neilconway.org/talks/kolmogorov.pdf
• http://people.cs.uchicago.edu/~fortnow/papers/quaderni.pdf
• http://c2.com/cgi/wiki?KolmogorovComplexity