Information Theory and Networks

Lecture 19: Complexity

Matthew Roughan
<matthew.roughan@adelaide.edu.au>
http://www.maths.adelaide.edu.au/matthew.roughan/Lecture_notes/InformationTheory/

School of Mathematical Sciences,
University of Adelaide

September 18, 2013

Part I

Complexity

For every problem there is a solution which is simple, clean and wrong.

Henry Louis Mencken

Simplicity and Occam's razor

> Pluralitas non est ponenda sine neccesitate
>
> *William of Ockham (ca. 1285-1349)*

- "Plurality should not be posited without necessity."
- alternative versions
  - "Entia non sunt multiplicanda praeter necessitatem", or "Entities should not be multiplied beyond necessity"
  - "in vain we do by many which can be done by means of fewer"
  - "if two things are sufficient for the purpose of truth, it is superfluous to suppose another"
  - Principle of Parsimony

Quidquid latine dictum sit, altum viditur.

Complexity

- Occam's Razor is often interpreted as "simple theories are best" (all else being equal)
- But what do "simple" or "complex" mean?
  - computational complexity
    - computational resources (e.g. CPU or memory) required by an algorithm
  - emergence and self-organization
    - e.g. flocking behaviour
    - e.g. Conway's game of life
    - e.g. consciousness
  - non-linearity and "chaos"
  - irreducible systems
    - systems that are more than the sum of their parts?
  - programming complexity
    - metrics for describing how complicated a computer program is
    - e.g., length of code, vocabulary,
    - e.g., count of linearly independent paths through the code

“complicated” vs “complex”

Warren Weaver, 1948

- **disorganised complexity:**
  - large number of relationships, often can be considered almost independent
  - “complicated” = lots of moving parts, but reducible to these
  - use probability and statistical mechanics to analyse, e.g., temperature of a gas, roll of a dice, ...
- **organised complexity:**
  - smaller (maybe still large) number of relationships, that can’t be treated as independent
  - non-random, but hard to predict
  - “complex” = small number of parts can generate “interesting” behaviour
  - analyse (typically) through simulation

Complexity

- Why do I care:
  - complex systems are harder to manage
  - how can we make them simpler if we don't even understand what that means
- We're interested in strings (signals or messages) so lets talk about them?

Complexity examples

- We're interested in strings (signals or messages) so lets talk about them?
- Which of these is complex?
  1. 101010101010101010101010101010101
  2. 110010010000111111011010101000100
  3. 101001010100001010101111010101010

Complexity examples

- We're interested in strings (signals or messages) so lets talk about them?
- Which of these is complex?
  1. 101010101010101010101010101010101
  2. 110010010000111111011010101000100
  3. 101001010100001010101111010101010
- Answers:
  1. repeat 10
  2. π in binary
  3. some random bits I typed

Kolmogorov Complexity

- The basic idea is that the complexity is the length of the shortest description of the sequence
  - "description" could mean a program to generate it
  - or it could just be "write the string 10101..."
- Obviously this is still a little vague
  - what programming language and computer?

Turing Machine

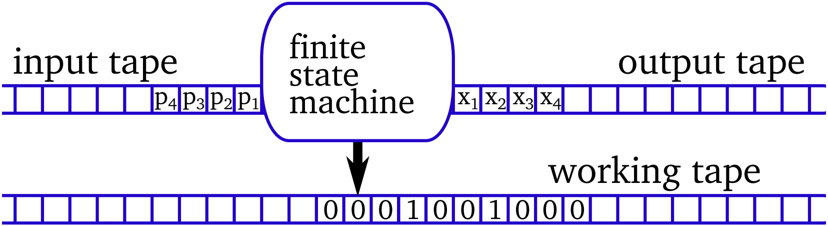
- An abstract model of a computer
- Turns out that all sufficiently complex computing systems are equivalent in the sense that they can compute the same family of functions:
  - computable functions intuitively have a finite program, that completes in a finite number of steps to the result
  - almost all functions we deal with in math are computable (though maybe not efficiently)
  - there are a few that aren't
- Turing machines have a few variants, but simplest has
  - a tape
  - a finite state machine that can write/read from the tape

Simple Turing Machine

- a tape
  - a **tape** is an idealisation of computer memory
  - imagine a strip of paper on which we can write or erase some symbols (often binary 1s and 0s)
  - the tape can be moved back and forth so that the machine can write and read any point on the tape
- a finite state machine that can write/read from each tape
  - *n* states, plus "halt"
  - transition function has inputs of current state and current tape value
  - transition causes three outputs:
    - can write over the current bit of the tape
    - it can move the tape
    - the state machine's state can change
- running the machine means setting a set of tape values, and a starting state, and then allowing transitions until "halt" is reached

Our Turing Machine

- Ours will be just a little different (but equivalent)



- Its helpful to separate inputs and outputs from working memory
  - input tape (with the input **p** – the program – on it)
  - output tape (which we will write the output **x** on)
  - a working tape
  - a finite state machine that can write/read from each tape
- We'll call this a universal computer

Formal Kolmogorov Complexity

Definition (Kolmogorov Complexity)

The Kolmogorov complexity $K_{\mathcal{U}}(\mathbf{x})$ of a string $\mathbf{x}$ with respect to a universal computer $\mathcal{U}$ is defined as

$$K_{\mathcal{U}}(\mathbf{x}) = \min_{\{\mathbf{p} | \mathcal{U}(\mathbf{p}) = \mathbf{x}\}} \ell(\mathbf{p})$$

So we are

- minising the length $\ell(\mathbf{p})$ of the input $\mathbf{p}$
- such that the output $\mathcal{U}(\mathbf{p}) = \mathbf{x}$
- and then it halts

Further reading I

Thomas M. Cover and Joy A. Thomas, *Elements of information theory*, John Wiley and Sons, 1991.