Chapter 3: Kolmogorov complexity

Intro

Sequence completion : how?
" probability ? preference?
" model ?

Find a model for the following sequences:

- 01010101010101010101 .... : repeat '01'
- 011010100000100111100110011001111111.... $\sqrt{2}-1$ in binary
- 110111001110101100111101101110110101 : too many '1' for uniform law

$-\frac{1}{2}\log\frac{1}{2}-\frac{1}{2}\log\frac{1}{2}=-\log\frac{1}{2}=+\log 2=1$

0 1
{0.5; 0.5}

↳ length (encoding) : $n \times H(p) = n$

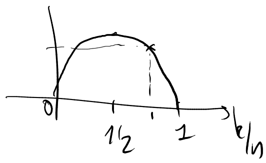
k = count number of 1s

bencode : $\log n + n H\left(\frac{k}{n}\right)$

↳ 0 1
p : $1-k/n$ k/n
$H(k/n) := H(p)$

or
order all sequences of length n with k 1s
then encode the index of this particular sequence

$H(k/n)$



English text + same one in German
→ compress each other independently : if short text
→ if very long (full library):
- encode how to translate English to German
- encode the English text only

I Kolmogorov complexity

Definition [Cover & Thomas, p 463]

- K(string s) = description cost of s = the length of the shortest program that can produce it
  ↳ unit: bits
  ↳ ex: s = 00000000 . . 00 (1000 of them) : repeat 1000 '0' → 15
  $0^\wedge 1000$ → 6
  $s = 3.14159$ . . (1000 first digits of π)
- Associated probability: $2^{-K(s)}$ → distribution over strings
  ↳ sum up to 1? no
  $\sum_s 2^{-K(s)} \to \Omega$: Chaitin's constant → provably non-computable
  ↳ halting problem

Universality [Cover & Thomas, p. 427]

- more or less Turing-machine-invariant
  → universal Turing machine
  → Church's thesis: all computational models are equivalent (if sufficiently complex)

$K_{\text{computer 1}}(s) \leq K_{\text{computer 2}}(s) + C_{1,2} \quad \forall s$

↳ by running a simulator of computer 2 (of size $C_{1,2}$) on computer 1 and executing the program associated to $K_{\text{computer 2}}(s)$

⇒ we'll always have + constant in all formulas bounding $K(s)$

- not dependent on encoding:
  - given 2 possible binary encodings f & g for data x (which is not binary)

$$K(g(x)) \leq K(f(x)) + K(\underbrace{g \circ f^{-1}}_{\text{translation from an encoding to the other one}}) \quad (+c^{st})$$

- these constants are small: < 1MB
  compared to possible big data size (GB, TB)
  ↳ Kolmogorov complexity really makes sense

Extensions (defined up to a multiplicative factor, e^{+cst})

P_1 - probability associated with Kolmogorov complexity: $p(s) = 2^{-K(s)}$

P_2 - other version: $\sum_{\text{all programs } f \text{ that produce } s} 2^{-\text{length}(f)}$

P_3 - with probabilistic programs: $\sum_{\text{all random programs } f} 2^{-\text{length}(f)} \times \underbrace{P(f \text{ outputs } s)}_{\text{probability}}$

P_4 - with distributions:

$$\sum_{\text{all proba distributions } \mu} 2^{-\text{length to describe}(\mu)} \times \mu(s)$$

Proposition: these 4 definitions are equivalent, i.e.: [Zvonkin & Levin, 1970]

$\forall$ definitions i, j, $\exists\, C \in \mathbb{R}$, $P_i \leq C\, P_j$

$\exists c_1, c_2 \geq 0$
$\forall s: c_1 P_1(s) \leq P_2(s) \leq c_2 P_1(s)$

↳ named Solomonoff universal prior (for prediction)

Relative complexity: $K(s|z)$ when z is already available → no need to describe it
↳ similar to mutual information, conditional entropy, etc.

II Bounds

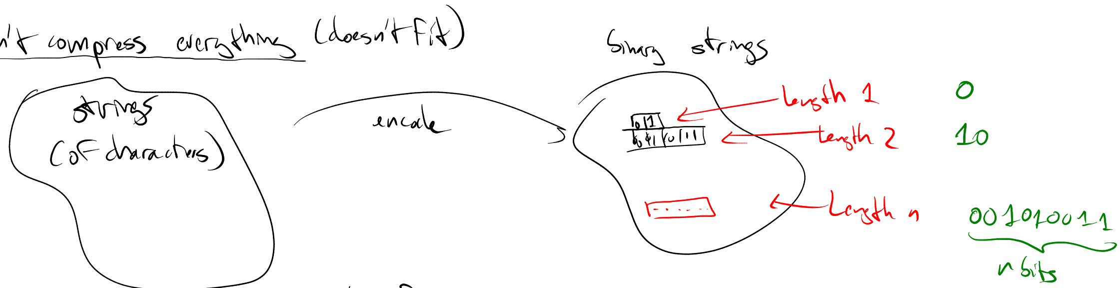
Easy upper bound

. $K(s) \leq \text{length}(s) + 2 \log \text{length}(s) + c$
or $+ \log + \log\log + \log\log\log + \dots + c$
(cf previous course)

program: "print the following chain, of length ___ : ___"
length(s) s

. $K(s) \leq |s| + K(\underbrace{|s|}_{\in \mathbb{N}}) + c$ more general

. Can't compress everything (doesn't fit)



number of strings s
s.t. $K(s) \leq n$
is $1 + 2 + 2^2 + 2^3 + 2^4 + \dots + 2^n < 2^{n+1}$
⇒ not many strings are simple

↳ strings s such that $K(s \mid |s|) \geq |s|$ are named "algorithmically random" by Kolmogorov
(because no regularity to exploit)

↳ infinite strings $s = (x_1, x_2, \dots)$ such that $\lim_{n\to\infty} \frac{K(x_1 x_2 \cdots x_n \mid n)}{n} = 1$ are called incompressible

. $K(s) \leq |zip(s)| + |\text{unzip program}|$ "Clustering by compression" [Cilibrasi & Vitanyi]

↳ distance based on zip used to cluster files (text, MIDI ...) → it worked!

↳ $d(x,y) = \frac{\max(K(x|y), K(y|x))}{\max(K(x), K(y))}$ using zip as a proxy for K

. If x is in $\mathcal{E}$ (finite set):

$$K(x) \leq K(\mathcal{E}) + \lceil \log |\mathcal{E}| \rceil + c$$

say which element of the set $\mathcal{E}$ x is

↳ generalization: given a set X (not necessarily finite) and a probability measure on it,

$$K(x) \leq K(\mu) - \log \mu(x) + c$$

↳ in machine learning: a model μ is good if this quantity is small!

. Theorem: Kolmogorov complexity is non computable!

- cf Gödel, Turing, halt problem → indecidability of the output of some programs
  ↳ whether they halt
  ↳ and if yes, what they output
- Cannot prove that $K(x) > 1\,MB$, whatever x is
  "Berry paradox":
  " The smallest number that cannot be described in less than 13 words. " → description of a number using 12 words
  "The first x found that cannot be described in less than L bits"
- Paradoxal proof in 2 steps, by Chaitin: Incompleteness theorem, 1971
  - step 1: proposition: $\exists L$ s.t. it is not possible to prove the statement: $K(x) > L$ for any x

    proof: $L = 1\,MB$.
    - write a program that goes through all possible proofs and stop when finding a proof of $K(x) > L$ for some x and print that x
    - this program has length $< L$
    - if it stops and prints an x ... x can be described by a program of length $< L$!

    ⇒ contradiction

    ⇒ this program never stops

    ⇒ there doesn't exist any proof of the form "$K(x) > 1\,MB$" for any x

  - step 2: theorem: Kolmogorov complexity is not computable

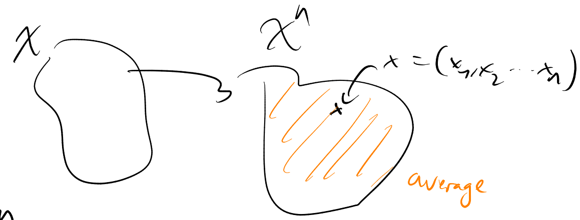
    proof: . consider all integers $\in [1, 2^{L+1}]$
    . there are at most $2^{L+1} - 1$ programs of length $\leq L$
    . $2^{L+1} > 2^{L+1} - 1 \Rightarrow \exists n_0 \in [1, 2^{L+1}]$ s.t. $K(n_0) > L$
    . if there existed a program able to compute $K(n)$ for any n, by computing $K(n_0)$ we would prove $K(n_0) > L$ ⇒ not possible ⇒ such a program does not exist.

$\Rightarrow$ not possible to have lower bounds on K (sufficiently complex string)
↳ provided it's over 1 MB

Entropic bound [Cover & Thomas, p 473]

→ cannot prove a lower bound of K for a given string s
but
can prove lower bounds on average

→ consider a proba distribution μ over a set $\mathcal{X}$



$$H(\mathcal{X}) \le \underbrace{\mathbb{E}_{x \sim \mu^n}\left[\tfrac{1}{n} K(x)\right]}_{\text{average } K \text{ complexity}} \le H(\mathcal{X}) + \underbrace{(|\mathcal{X}|+1)\frac{\log n}{n} + {}^{c}/_{n}}_{\xrightarrow[n\to\infty]{} 0}$$

$$\Rightarrow \boxed{\lim_{n\to\infty} \mathbb{E}_{x\sim\mu^n}\left[\tfrac{1}{n}K(x)\right] = H(\mathcal{X})}$$

cannot do better than entropy (on average)

Proof: lower bound:
- $K(s)$ is the length of an encoding $\Rightarrow$ Gibbs inequality $\Rightarrow$ cannot be better than entropy ($KL \ge 0$)

upper bound: by explicit construction → write a program outputting s and check its length ($\ge K(s)$)
- encode n: costs $2\log n$
- string $s = x = (x_1, x_2 \dots x_n)$ ← symbols from an alphabet $\mathcal{A} = \{'a', 'b', \dots 'z'\}$
  ↳ count the # of appearance of each symbol in s ⟶ $n_a \quad n_b \quad n_z$
- encode them:
  cost: $\underbrace{|\mathcal{A}|}_{|\mathcal{X}|} \log n$, actually $(|\mathcal{A}|-1)$
- now: draw the list of all possible sequences of n characters with exactly n_a 'a', n_b 'b', ...
  ↳ this list: size $\le 2^{nH(\text{Bernoulli}(n_a/n,\, n_b/n, \dots))}$
  ↳ $C_k^n = \binom{n}{k} \le 2^{nH(\dots)}$ using Stirling formula
  ↳ $\sum_k \underbrace{\binom{n}{k} p^k (1-p)^{n-k}}_{\le 1} = 1$ with $p = {}^{k}/_{n}$
- encode the index of s within this list
  ↳ $\le n H(\mathcal{X})$ using Jensen

$\Rightarrow$ total encoding cost for s = what is written in the formula

III Minimum Length Description principle (MDL)

Def° General criterion for model selection
Given a set $\mathcal{X}$ and a probability distribut° μ on it,

Encoding cost w.r.t. μ:
$-\log p(x)$
$-\log \mu(x)$

for any x, $\quad K(x) \le \underbrace{K(\mu)}_{\text{complexity of the model}} + \underbrace{-\log \mu(x)}_{\text{Likelihood: how well the model fits the data}} \quad (+c)$

= natural trade-off between model complexity and accuracy → pick the model μ with the lowest $K(\mu) - \log \mu(x)$

Occam's razor

Unsupervised learning task

dataset $\mathcal{D}$, goal: generate new points according to $\mathcal{D}$

$\mathcal{D} = \{x_i\}$ observed points

↳ generator G → defines a probability p_G over $\mathcal{X}$

model

μ

$$K(\mathcal{D}) \leq K(G) \underbrace{- \sum_i \log p_G(x_i)}_{-\log \prod_i p_G(x_i) \,=\, -\log p_G(\mathcal{D})}$$

Supervised learning task

training dataset $\mathcal{D} = \{(x_i, y_i)\}$

↳ space $\mathcal{X} \times \mathcal{Y} =: \mathcal{Z}$

Goal: a function $\mathcal{X} \to \mathcal{Y}$

$x \longmapsto y$

deterministic view

graph of the function

proba measure defined on $\mathcal{Z}$

↳ $p_M((x,y))$? $\quad p_M(y|x)\, p_M(x)$

Function

non-deterministic view

$$K(\mathcal{D}) \leq K(p_M) \underbrace{- \log p_M(\mathcal{D})}_{-\log \prod_{i \in \mathcal{D}} p_M((x_i, y_i))}$$

$$= \sum_i -\log p_M((x_i, y_i))$$

space of all possible datasets

Examples

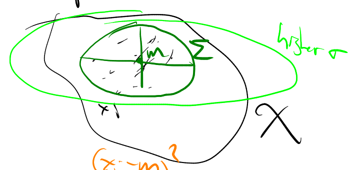
- Dirac peak $\mu = \delta_{\mathcal{D}}$

↳ $\underbrace{K(\mu)}_{K(\mathcal{D})} \underbrace{-\log \mu(\mathcal{D})}_{0}$

overfit: high model complexity, high likelihood

- Gaussian distribution $\mathcal{N}(m, \sigma)$, fit to a cloud of points $\mathcal{D} = \{x_i\}$

$\sigma \to \Sigma$



$K(\mu) - \log \mu(\mathcal{D})$

$K(\mu) \to K(m),\ K(\sigma)$

↓

encode 2 real values to encode

↓

up to which precision?

$m = 1{,}205|72301...$

trade off between the precision of m encoded & the accuracy of the model $\mathcal{N}(m, \sigma)$

$-\log \mu(\mathcal{D}) \to -\sum_i \log p_\mu(x_i)$

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i - m)^2}{2\sigma^2}} \quad \text{(in dimension 1)}$$

$$\sum_i + \frac{1}{2}\log(2\pi\sigma) + \frac{(x_i - m)^2}{2\sigma^2}$$

data term: $\sum_i (x_i - m)^2$

+ regularizer

factor?

↘ given by MDL (no need to search for it)

↳ MDL is a general approach to define ML problems i.e. to set up the training criterion

Model selection:

given N models μ_k, select model μ_{k^*} that minimizes $K(\mu) - \log \mu(\mathcal{D})$

Instanciations

- AIC, BIC : approximations of K (model)
  - ↳ AIC: Akaike Information Criterion [1973]

    $K(\mu) :=$ number of parameters of the model μ

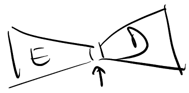
  - ↳ BIC: Bayesian Information Criterion [Schwartz, 1978]

    $K(\mu) := \frac{1}{2}$ number of parameters $\times$ Log(number of observations)

    ↳ dataset size

Restricted families of programs

- $K(s) := |zip(s)|$
- auto-encoders : the middle layer = compressed data



- programs on Turing machines
  → restrict to finite automata ⇒ Kolmogorov complexity is computable

IV Conclusion

- MDL = very general principle, to formulate any Machine Learning problem