Advanced topics in AI / Foundations of AI
Assignment 1 Report

Zhao, Longfei u5976992
Luo, Dong u5319900

September 21, 2017

Total time spent: about 30 hours each student

1 PI-Seq

(a) $x_3 = 3$, since 3 is the next natural number. x_3 may also be 4, if $x_i = 2^{i-1}$. But 3 is preferred as the sequence of natural numbers is such a simple sequence.

(b) $x_5 = 5$, since 3 is the next natural number. x_5 may also be 0 as the sequence might be defined as $x_i = imod5$

(c) 61, since it is the next prime. It can also be 60, supposing the sequence represents the street numbers of shops on one side of a street.

(d) 5, as it is the next digit of π. It might also be 3 or 1 as they have higher frequencies in the first ten numbers.

No matter how many numbers we have, there will not be sure of continuation, though we have more information when we know more numbers. For example, when we have a sequence whose first 1 million numbers are all 1's, it is still possible that the sequence is defined as $x_i = 1, \forall i \leq 1,000,000$ and $x_i = i, \forall i > 1,000,000$.

2 IT-Id

(i) We can construct a bijection $f : \mathbb{B}^* \to \mathbb{N}$ as follows:
Let x be a binary string, we append a 1 to the left of and x and treat it as a natural number in binary representation, then define $f(x) = (1x - 1)_2$. For example, $f(\epsilon) = (1 - 1)_2 = 0_{10}$ and $f(0) = (10 - 1)_2 = 1_{10}$.
f defined in this way is a bijection. f is injective since if x and y are different strings, $1x$ and $1y$ are different numbers (in binary representation), and it follows that $1x - 1 \neq 1y - 1$.
f is surjective since any natural number (including 0) n, after adding 1, is larger or equal to 1, and must has its binary representation starting with 1. If we treat $(n + 1)_2$ as a binary string and remove the first 1, we got a binary string x and by our construction of x, we must have

$f(x) = (1x-1)_2 = (n+1-1)_2 = n_2$.

f is a bijection and can be computed in polynomial time.

(ii) $\overline{x} = 1^{l(x)}0x$, then $l(\overline{x}) = 2l(x)+1$

Now prove $\overline{x}$ is prefix-free,

Assume $x \neq y$ and $\overline{x}$ is $\overline{y}$'s prefix, then

$\exists z : 1^{l(x)}0xz = 1^{l(y)}0y$

$\Rightarrow l(x) = l(y)$

$\Rightarrow xz = y$

$\Rightarrow z = \emptyset$

Therefore, $\overline{x}$ is prefix-free

(iii) $x' = \overline{l(x)}x$, then $l(x') = l(x) + 2l(l(x)) + 1$

$\log(x+1) - 1 < l(x) \leq \log(x+1)$

$\Rightarrow l(x') \leq \log(x+1) + 2\log(\log(x+1)+1) + 1 \sim \log x + 2\log\log x$

Now prove x' is prefix-free,

Assume $x \neq y$ and x' is y''s prefix, then

$\exists z : \overline{l(x)}xz = \overline{l(y)}y$

we know $\overline{x}$ is prefix-free

$\Rightarrow \overline{l(x)} = \overline{l(y)}$

$\Rightarrow l(x) = l(y)$

$\Rightarrow xz = y$

$\Rightarrow z = \emptyset$

Therefore, x' is prefix-free

(iv) According (ii) and (iii), $l(\overline{x}) = 2l(x)+1$, $l(x') = l(x) + 2l(l(x)) + 1$, then

$\Rightarrow l(\overline{x}) - l(x') = l(x) - 2l(l(x)) \sim \log x - 2\log\log x$ so, approximately when $x > 16$, $\overline{x}$ is longer than x', when $x < 16$, $\overline{x}$ is shorter than x'

(v) Pseudocode as follow. Assume there is a header of this input stream.

Algorithm 1 Decode first-order prefix coding to original coding

Input: The header of this half-infinite first-order prefix binary string, which start with $\overline{x}$
Output: Original binary string x

```
1:  function GetOriginalFromFirstOrderPrefix(header)
2:      length ← 0
3:      pointer ← header
4:      while Value(pointer) ≠ 0 do          ▷ get the value at pointer position
5:          length ← length + 1
6:          pointer ← pointer + 1
7:      end while
8:      start ← pointer + 1
9:      end ← start + length
10:     return GetString(start, end)         ▷ get values from start position to end position
                                               and combine them to a binary string
11: end function
```

(vi) Already prove those in (ii) and (iii)

Algorithm 2 Decode second-order coding to original coding

Input: The header of this half-infinite second-order prefix binary string, which start with x‘
Output: Original binary string x

function GetOriginalFromSecondOrderPrefix($header$)
    $length \leftarrow 0$
    $pointer \leftarrow header$
    while Value($pointer$) $\neq 0$ **do** ▷ get the value at $pointer$ position
        $length \leftarrow length + 1$
        $pointer \leftarrow pointer + 1$
    end while
    $start \leftarrow pointer + 1$
    $end \leftarrow start + length$
    $lengthString \leftarrow$ GetString($start, end$) ▷ get values from $start$ position to end position and combine them to a binary string
    $length \leftarrow$ GetNumberFromBinaryString($lengthString$) ▷ get the number corresponding to this binary string
    $start \leftarrow end + 1$
    $end \leftarrow start + length$
    return String($start, end$) ▷ get values from $start$ position to end position and combine them to a binary string
end function

3 KC-KC

(i) We'll need to prove three propositions in this question.

Proposition 3.1. $\sum_{x\in\{0,1\}*} 2^{-K(x)} \leq 1$

Proof. For each $x \in \{0,1\}*$, there exist a program describes x, since a finite binary string must be computable (Theorem 2.11). Let p_x denote the program of the shortest length such that $U(p_x) = x$. By definition, $\ell(p_x) = K(x)$. Therefore

$$\sum_{x\in\{0,1\}*} 2^{-K(x)} = \sum_{x\in\{0,1\}*} 2^{-\ell(p_x)}. \tag{1}$$

As p_x is a self-deliminating program for all x, the Kraft's inequality gives

$$\sum_{x\in\{0,1\}*} 2^{-\ell(p_x)} \leq 1. \tag{2}$$

Hence, we have

$$\sum_{x\in\{0,1\}*} 2^{-K(x)} \leq 1. \tag{3}$$

□

Proposition 3.2. $K(n) \to \infty$ *for* $n \to \infty$

Proof. We can show that $\forall M \in \mathbb{N}, \exists N$, such that $\forall n > N, K(n) > M$. There are only $\sum_{i=0}^{M} 2^i = 2^{M+1} - 1$ different binary strings with length less or equal to M. These strings can be descriptions of at most $2^{M+1} - 1$ numbers. Let $N = \max\{x : x \in \mathbb{N}K(x) \leq M\}$. N is finite since there are at most $2^{M+1} - 1$ x such that $K(x) \leq M$. By our construction,

$\forall n > N, K(n) > M$.
By definition, $K(n) \to \infty$ for $n \to \infty$. □

Proposition 3.3. *$K(x) \geq \ell(x)$ for 'most' x. 'most' means there are only $o(N)$ exceptions for $x \in \{1, \ldots, N\}$.*

Proof. Define $f(s) = |\{n : \ell(n) = s, K(n) < \ell(n)\}|$ (the number of compressible strings of length s),
$f'(s) = \sum_{i=1}^{s} f(i)$ (the number of compressible strings of length less or equal to s),
$g(s) = |\{n : \ell(n) = s\}| = 2^s$ (number of strings with length equal to s), and
$g'(s) = |\{n : \ell(n) \leq s\}| = 2^{s+1} - 1$ (number of strings with length less or equal to s).
By Theorem 3.3.1 on *An Introduction to Kolmogorov Complexity and its Applications*, consider strings of length n, for any integer r, the number of strings x such that $K(x) \leq n + K(n) - r$ is less than $2^{n-r+O(1)}$.
Set $r = K(n)$, we find that

$$|\{x : K(x) \leq n\}| \leq 2^{n-K(n)+O(1)},$$

where $n = \ell(x)$.
We notice that as n goes to infinity, the proportion of 'compressible strings' of length n goes to infinity.

$$\frac{f(n)}{g(n)} \leq \frac{2^{n-K(n)+O(1)}}{2^n} = 2^{-K(n)+O(1)},$$

which goes to 0 as $n \to \infty$. As $K(n) \to \infty$ for $n \to \infty$, $\lim_{n\to\infty} 2^{-K(N)+O(1)} = 0$.
$\exists C$, such that $2^{-K(C)+O(1)} < \frac{\epsilon}{2}, \forall \epsilon$.
Since $\forall C' \in \mathbb{N}$, there are finitely many $n \in \mathbb{N}$ such that $K(n) < C'$. Let $n' = \max\{n : n \in \mathbb{N}, K(n) < K(C)\}$.
By the definition of n', $\forall n > n', K(n) > K(C)$. Hence,

$$2^{-K(n)+O(1)} < 2^{-K(C)+O(1)} < \frac{\epsilon}{2}.$$

Now consider $M > n'$.

$$g'(M) = 2^{M+1} - 1,$$

and

$$\begin{aligned} f'(M) &= \sum_{i=1}^{M} f(i) \\ &= \sum_{i=1}^{n'} f(i) + \sum_{i=n'+1}^{M} f(i) \\ &= f'(n') + \sum_{i=n'+1}^{M} f(i) \\ &\leq g'(n') + \sum_{i=n'+1}^{M} 2^i \frac{\epsilon}{2} \\ &\leq g'(n') + \frac{\epsilon}{2} 2^{M+1} \end{aligned}$$

$\forall \epsilon$, when $M > \log_{(}\frac{2^{n'+1}}{\epsilon})$, we have $\frac{f'(M)}{g'(M)} < \epsilon$. That means as n goes to infinity, the proportion of compressible strings among all strings of length less or equal to n goes to 0.

Finally we are able to prove the proposition using what we just showed. Let $\{1, \ldots, N\}$ be a set of natural numbers (using the normal binary string representation), we want to show that when N is goes to ∞, the ratio of n such that $K(n) < \ell(n)$ goes to 0.

Let

$$h(N) = |\{n : n < N, K(n) < \ell(n)\}|,$$

(the number of n's in $\{1, \ldots, N\}$ such that $K(n) < \ell(n)$).

Suppose $\ell(N) = s$, then $g'(s-1) < N \leq g'(s)$ since $g'(s)$ is the number strings with length less or equal to s.

$$\frac{h(N)}{N} = \frac{h(g'(s-1))}{N} + \frac{h(N) - h(g'(s-1))}{N}.$$

We find that $\frac{h(g'(s-1))}{N} = \frac{f'(s-1)}{N} \leq \frac{f'(s-1)}{g'(s-1)}$, and we've shown that $\frac{f'(s-1)}{g'(s-1)}$ goes to 0 as s goes to ∞, but as $s = \ell(N) \to \infty$ when $N \to \infty$, $\frac{f'(s-1)}{g'(s-1)} \to 0$ as $N \to \infty$.

The other part

$$\frac{h(N) - h(g'(s-1))}{N} \leq \frac{h(N) - h(g'(s-1))}{g'(s-1)} = \frac{h(N) - h(g'(s-1))}{g(s) - 1} \leq \frac{f(s)}{g(s) - 1},$$

which also converges to 0 when $s \to \infty$.

To conclude,

$$\lim_{N\to\infty} \frac{h(N)}{N} = \lim_{N\to\infty} \frac{h(g'(s-1))}{N} + \lim_{N\to\infty} \frac{h(N) - h(g'(s-1))}{N} = 0.$$

□

(ii) Prove $K(x|y) \overset{+}{<} K(x)$,

Let p with a TM T_{i_1} be shortest description of x under U, then

$T_{i_1}(p) = x, U(i_1 `p) = x, K(x) = l(i_1`) + l(p)$

$\exists i_2 : T_{i_2}(y`p) = T_{i_1}(p) = x$, i.e. $T_{i_2}$ can add a process to ignore the side information $y$ based on $T_{i_1}$, then $l(i_1) \overset{+}{=} l(i_2) \Rightarrow l(i_1`) \overset{+}{=} l(i_2`)$

$\Rightarrow U(y`i_2`p) = U(i_1`p) = x$

$\Rightarrow K(x|y) \leq l(i_2`p) = l(i_2`) + l(p) \overset{+}{=} l(i_1`) + l(p) = K(x)$

$\Rightarrow K(x|y) \overset{+}{<} K(x)$

Prove $K(x) \overset{+}{<} K(x, y)$

Let p with a TM T_{i_1} be shortest description of $x`y$ under U, then

$T_{i_1}(p) = x`y, U(i_1`p) = x`y, K(x, y) = K(x`y) = l(i_1`) + l(p)$

$\exists i_2 : T_{i_2}(p) = x$, i.e. T_{i_2} can add a process to halt when output x based on T_{i_1}, then $l(i_1) \overset{+}{=} l(i_2) \Rightarrow l(i_1`) \overset{+}{=} l(i_2`)$

$\Rightarrow U(i_2`p) = x$

$\Rightarrow K(x) \leq l(i_2`p) = l(i_2`) + l(p) \overset{+}{=} l(i_1`) + l(p) = K(x, y)$

$\Rightarrow K(x) \overset{+}{<} K(x, y)$

Therefore, $K(x|y) \overset{+}{<} K(x) \overset{+}{<} K(x, y)$

(iii) Prove $K(xy) \overset{+}{<} K(x,y)$

Let p with a TM T_{i_1} be shortest description of $x`y$ under U, then
$T_{i_1}(p) = x`y, U(i_1`p) = x`y, K(x,y) = K(x`y) = l(i_1`) + l(p)$
Define $r :=$ decode prefix code program which is a constant length, then
$\Rightarrow U(r`i_1`p) = xy$
$\Rightarrow K(xy) \leq l(r`i_1`p) \overset{+}{=} l(i_1`) + l(p) = K(x,y)$

Prove $K(x,y) \overset{+}{<} K(x) + K(y|x)$
Let p_1 with a TM T_{i_1} be shortest description of x and p_2 with a TM T_{i_2} be shortest description of $x|y$ under U, then
$T_{i_1}(p_1) = x, U(i_1`p_1) = x, K(x) = l(i_1`) + l(p_1)$
$T_{i_2}(x`p_2) = y, U(x`i_2`p_2) = y, K(y|x) = l(i_2`) + l(p_2)$
Define $r :=$ code prefix code program which is a constant length, then
$\Rightarrow U(r`i_1`p_1`i_2`p_1) = x`y$
$\Rightarrow K(xy) \leq l(r`i_1`p_1`i_2`p_1) \overset{+}{=} l(i_1`) + l(p_1) + l(i_2`) + l(p_2) = K(x) + K(y|x)$
$\Rightarrow K(xy) \overset{+}{<} K(x) + K(y|x)$

Prove $K(x) + K(y|x) \overset{+}{<} K(x) + K(y)$
according to (ii), $K(x|y) \overset{+}{<} K(x)$
$\Rightarrow K(x) + K(y|x) \overset{+}{<} K(x) + K(y)$
par Therefore, $K(xy) \overset{+}{<} K(x,y) \overset{+}{<} K(x) + K(y|x) \overset{+}{<} K(x) + K(y)$

(iv) f is computable $\Rightarrow \exists i_1 : T_{i_1}(x) = f(x)$ and $K(f) = i_1$, then
$\Rightarrow U(i_1) = f$
Let p with a TM T_{i_2} be shortest description of x under U, then
$\Rightarrow T_{i_2}(p) = x, U(i_2`p) = x, K(x) = l(i_2`) + l(p)$
$\Rightarrow U(i_1`i_2`p) = f(x)$
$\Rightarrow K(f(x)) \leq l(i_1`i_2`p) \overset{+}{=} l(i_2`) + l(p) + l(i_1) = K(x) + K(f)$

(v) x is complex $\Rightarrow K(x) \to \infty$
$K_{u'}(x) = 1 \Rightarrow c_{uu'} \geq |K_u(x) - K_{u'}(x)| \to \infty$
We know, for any two natural Turing machine, the $c_{uu'}$ should be small $\Rightarrow U'$ is not a natural Turing machine.

(vi) Prove $K(n) \leq \log_2 n + O(\log\log n)$
There exists a TM T_i that $T_i(x`) = x$, then
$\Rightarrow U(i`x`) = x$
$\Rightarrow K(x) \leq l(i`x`) \overset{+}{=} l(x`) \overset{+}{<} l(x) + 2\log l(x)$
$\Rightarrow K(n) \overset{+}{<} \log n + 2\log\log n$
$\Rightarrow K(n) \overset{+}{<} \log n + O(\log\log n)$

Prove $K(0^n) \overset{+}{=} K(1^n) \overset{+}{=} K(n \text{ digits of } \pi) \overset{+}{=} K(n)$
Let p with a TM T_j be shortest description of n under U, then
$T_j(p) = n, U(j`p) = n, K(n) = len(j`) + len(p)$
Define $r_1 :=$ output n digits of 0, given n
Define $r_2 :=$ output n digits of 1, given n

Define $r_3 :=$ produce π and output n digits of π, given n
$\Rightarrow U(r_1`j`p) = 0^n, U(r_2`j`p) = 1^n, U(r_1`j`p) = n$ digits of π
$\Rightarrow K(0^n) \overset{+}{<} K(n), K(1^n) \overset{+}{<} K(n), K(n \text{ digits of } \pi) \overset{+}{<} K(n)$

In turn, we can easily assume three TMs $T_{i_1}, T_{i_2}, T_{i_3}$ with p_1, p_2, p_3 to compute $0^n, 1^n, n$ digits of π
Define $r_1 :=$ calculate the number of 0, given 0^n
Define $r_2 :=$ calculate the number of 1, given 1^n
Define $r_3 :=$ calculate the digits, given n digits of π
$\Rightarrow U(r_1`i_1`p_1) = n, U(r_2`i_2`p_2) = n, U(r_3`i_3`p_3) = n$
$\Rightarrow K(n) \overset{+}{<} K(0^n), K(n) \overset{+}{<} K(1^n), K(n) \overset{+}{<} K(n \text{ digits of } \pi)$
$\Rightarrow K(0^n) \overset{+}{=} K(1^n) \overset{+}{=} K(n \text{ digits of } \pi) \overset{+}{=} K(n)$

(vii)

4 KC-Cmp

(i) We will first show that if f is estimable, it must be upper and lower semicomputable.
Suppose f is estimbale, $\exists$ recursive $\phi, \forall x, \forall \epsilon > 0, |\phi(x, \frac{1}{\lfloor \epsilon \rfloor}) - f(x)| < \epsilon$
Then $\forall x$, we define $\phi^L(x,t) = \phi(x, \frac{1}{\lfloor \frac{1}{t} \rfloor}) - \frac{1}{t}$, and $\phi^U(x,t) = \phi(x, \frac{1}{\lfloor \frac{1}{t} \rfloor}) + \frac{1}{t}$. ϕ^L and ϕ^U are recursive. As

$$|\phi(x, \frac{1}{\lfloor \frac{1}{t} \rfloor}) - f(x)| < \frac{1}{t}, \tag{4}$$

We have

$$-\frac{1}{t} < \phi(x, \frac{1}{\lfloor \frac{1}{t} \rfloor}) - f(x) < \frac{1}{t}. \tag{5}$$

We then have $\phi(x, \frac{1}{\lfloor \frac{1}{t} \rfloor}) + \frac{1}{t} > f(x)$, and $\phi(x, \frac{1}{\lfloor \frac{1}{t} \rfloor}) - \frac{1}{t} < f(x)$,
which is equivalent to $\phi^U(x,t) > f(x)$, and $\phi^L(x,t) < f(x)$.
The inequality 4 shows that

$$\lim_{t\to\infty} \phi(x, \frac{1}{\lfloor \frac{1}{t} \rfloor}) - f(x) = \lim_{t\to\infty} [\phi(x, \frac{1}{\lfloor \frac{1}{t} \rfloor}) - f(x)] = 0, \tag{6}$$

Therefore, we see that

$$\begin{aligned} \lim_{t\to\infty} \phi^L(x,t) - f(x) &= \lim_{t\to\infty} [\phi(x, \frac{1}{\lfloor \frac{1}{t} \rfloor}) - \frac{1}{t}] - f(x) \\ &= \lim_{t\to\infty} \phi(x, \frac{1}{\lfloor \frac{1}{t} \rfloor}) - \lim_{t\to\infty} \frac{1}{t} - f(x) \\ &= \lim_{t\to\infty} \phi(x, \frac{1}{\lfloor \frac{1}{t} \rfloor}) - f(x) \\ &= 0, \end{aligned} \tag{7}$$

We have $\lim_{t\to\infty} \phi^L(x,t) = f(x)$, and similarly $\lim_{t\to\infty} \phi^U(x,t) = f(x)$.
Denote $a_n = \phi^L(x,n), \forall n \in \mathbb{N}$, and let (a_n) be the sequence $(a_1, a_2, ...)$. By equation 7, the sequence (a_n) converges to $f(x)$.

We now claim that (a_n) has a monotonic increasing subsequence. Furthermore, the subsequence is computable.
As ϕ^L is computable, a_i is computable for all $i \in \mathbb{N}$. We can design a Turing machine which computes $a_1, a_2, a_3, ...$ We let $a_{n_1} = a_1$, then let a_{n_2} be the first a_i such that $a_i > a_{n_1}$. This is feasible since (a_n) converges to $f(x)$, and $a_i < f(x)$ for all i, if there exist some element a_i which is greater or equal to all elements after it, $|f(x) - a_j| > |f(x) - a_i| > 0$ for any $j > i$, this contradicts with the fact that (a_n) converges to $f(x)$. Therefore, we know that for any a_i, there exist some j, such that $j > i$, and $a_j > a_i$. We can continue this process: for any a_{n_i}, we let $a_{n_{i+1}}$ to be the first element in (a_n) after a_{n_i} which is greater than a_{n_i}. Since we can always find the next element of the subsequence using the method described, and $n_i < n_j$ if $i < j$, $(a_{n_i}) = (a_{n_1}, a_{n_2}, a_{n_3}, ...)$ is a monotonic subsequence of (a_n).
As a_{n_i} is a subsequence of (a_n), a_{n_i} converges to $f(x)$. For any x, we define $a_i = \phi^L(x, i)$, and let $\phi^{L'}(x, t) = a_{n_i}$, where a_{n_i} can be computed by the method described above. We have a recursive $\phi^{L'}$ such that $\lim_{t\to\infty} \phi^{L'}(x, t) = f(x)$ and $\phi^{L'}(x, t) \leq \phi^{L'}(x, t + 1)$. By definition, $f(x)$ is lower semicomputable.
Similarly, $f(x)$ is upper semicomputable.

We now want to show that if $f(x)$ is lower and upper semicomputable, then $f(x)$ is estimable. Suppose $f(x)$ is lower and upper semicomputable, then by definition there exist recursive ϕ^L, and ϕ^U, such that

$$\lim_{t\to\infty} \phi^L(x, t) = \lim_{t\to\infty} \phi^U(x, t) = f(x), \tag{8}$$

$$\phi^L(x, t) \leq \phi^L(x, t + 1), \tag{9}$$

and

$$\phi^U(x, t) \geq \phi^U(x, t + 1). \tag{10}$$

We claim that $\phi^L(x, t) \leq f(x)$ for all t. Suppose that there exist a t' such that $\phi^L(x, t') > f(x)$, by inequality 9 we must have $\phi^L(x, t) \geq \phi^L(x, t')$ for all $t > t'$. Hence $|\phi^L(x, t) - f(x)| > |\phi^L(x, t') - f(x)|$, which contradicts equation 8. Similar argument shows that $\phi^U(x, t) \geq f(x)$. We have

$$|\phi^L(x, t) - f(x)| = f(x) - \phi^L(x, t), \tag{11}$$

and

$$|\phi^U(x, t) - f(x)| = \phi^U(x, t) - f(x), \tag{12}$$

Let $\epsilon > 0$ be arbitrary, we know from the equation 8 that $\exists N_l \in \mathbb{N}$ such that $\forall t > N_l, f(x) - \phi^L(x, t) = |\phi^L(x, t) - f(x)| < \frac{\epsilon}{2}$, and $\exists N_u \in \mathbb{N}$, such that $\forall t > N_u, \phi^U(x, t) - f(x) = |\phi^U(x, t) - f(x)| < \frac{\epsilon}{2}$. Then if we let $N = \max\{N_l, N_u\}$, $\forall t > N$,
$\phi^U(x, t) - \phi^L(x, t) = [f(x) - \phi^L(x, t)] + [\phi^U(x, t) - f(x)] < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon$. We've shown that

$$\forall \epsilon, \exists N \in \mathbb{N}, \text{such that} \forall t > N, \phi^U(x, t) - \phi^L(x, t) < \epsilon. \tag{13}$$

As we have

$$\begin{aligned} |\phi^L(x,t) - f(x)| &= f(x) - \phi^L(x,t) \\ &< f(x) - \phi^L(x,t) + [\phi^U(x,t) - f(x)] \\ &= \phi^U(x,t) - \phi^L(x,t), \end{aligned} \tag{14}$$

when $\phi^U(x,t) - \phi^L(x,t) < \epsilon$, we must have $|\phi^L(x,t) - f(x)| < \epsilon$. Therefore, we can design a Turing machine M to compute the desired $\phi(x, \lfloor \frac{1}{\epsilon} \rfloor)$. Suppose M has x and $p = \lfloor \frac{1}{\epsilon} \rfloor$ as input, as $\epsilon > 0$, $\frac{1}{p+1} > 0$. M compute $\phi^L(x,t)$ and $\phi^U(x,t)$ for $t = 1, 2, 3, ...$ until t reaches a t' such that $\phi^U(x,t') - \phi^L(x,t') < \frac{1}{p+1}$. We've proved the existence of such t'. The Turing machine M just output $\phi^L(x,t')$. As

$$|\phi^L(x,t') - f(x)| < \frac{1}{p+1} = \frac{1}{\lfloor \frac{1}{\epsilon} \rfloor + 1} < \frac{1}{\left(\frac{1}{\epsilon}\right)} = \epsilon, \tag{15}$$

the Turing machine compute the ϕ such that $\forall \epsilon > 0, |\phi(x, \frac{1}{\lfloor \epsilon \rfloor}) - f(x)| < \epsilon$. By definition, $f(x)$ is estimable.

(ii) Suppose f(x) is recursive. Then we can define $\phi(x, \lfloor \frac{1}{\epsilon} \rfloor) = f(x), \forall x \in \mathbb{N}, \forall \epsilon > 0$. ϕ is recursive since f is recursive. Then $|\phi(x, \lfloor \frac{1}{\epsilon} \rfloor) - f(x)| = 0 < \epsilon, \forall \epsilon > 0$. By definition, f is estimable.
We've shown in the above section that if f is estimable then it is lower semicomputable.
If f is lower semicomputable, then there exist a recursive $\phi(\cdot, \cdot)$, such that $\lim_{t \to \infty} \phi(x,t) = f(x)$ and $\phi(x,t) \leq \phi(x, t+1)$. By definition, f is approximable.

(iii) We'll show that the concepts on both sides of the three implications $\Downarrow$ are different. It follows thatthe four concepts are pairwise different.

recursive $\neq$ estimable Let f be a function such that $f(x) = \sqrt{2}, \forall x \in \mathbb{N}$. Since $\sqrt{2}$ is irrational, it cannot be written as $\frac{p}{q}$ where p and q are integers. Hence f is not finitely computable.
However, using a simple bisection method we can estimate $\sqrt{2}$ to any degree of accuracy. It follows that f is estimable.

estimable $\neq$ lower semicomputable By Theorem 4.4, M is enumerable but not estimable.

enumerable $\neq$ approximable We can prove this by contradiction.
Suppose any approximable function is enumerable. Since M is enumerable, M is approximable. $\exists$ recursive ϕ and $\lim_{t \to \infty} \phi(x,t) = M(x)$. Then $\lim_{t \to \infty} -\phi(x,t) = -M(x)$. As $-\phi$ is (can be constructed) recursive, $-M$ is approximable.
By our assumption, $-M$ is lower semicomputable. It follows that M is upper semicomputable. Since M is also lower semicomputable, M is estimable. But we know from Theorem 4.4 that M is not estimable, we have a contradiction.
Our assumption cannot be true. Therefore, there exist approximable but not enumerable functions.

5 BP-CP

If $p(\cdot)$ satisfies the Kolmogorov axioms, then

- $p(\Omega|C) = \frac{p(\Omega \cap C)}{p(C)} = \frac{p(C)}{p(C)} = 1$
- $p(\{\}|C) = \frac{p(\{\} \cap C)}{p(C)} = \frac{0}{p(C)} = 0$
- $p(A \cup B|C) = \frac{p((A \cup B) \cap C)}{p(C)} = \frac{p((A \cap C) \cup (B \cap C))}{p(C)} = \frac{p(A \cap C) + p(B \cap C) - p(A \cap B \cap C)}{p(C)} = p(A|C) + p(B|C) - p(A \cap B|C)$
- For a decreasing sequence $A_1 \supset A_2 \supset A_3 ...$ of events with $\bigcap_n A_n = \{\}$, $\lim_{n \to \infty} p(A_n|C) = \frac{p(A_n \cap C)}{p(C)} = \frac{0}{p(C)} = 0$

Therefore, $p(\cdot|C)$ also satisfies the Kolmogorov axioms.

6 BP-Med

We know,

$$P(H_{diseased}) = 1\%$$

$$P(H_{healthy}) = 99\%$$

$$P(D_{positive}|H_{diseased}) = 99\%$$

$$P(D_{positive}|H_{healthy}) = 1\%$$

According to Bayes' theorem,

$$P(H|D) = \frac{P(H)P(D|H)}{\sum_i P(D|H_i)P(H_i)}$$

we can easily get,

$$\begin{aligned} & P(H_{diseased}|D_{positive}) \\ =& \frac{P(H_{diseased})P(D_{positive}|H_{diseased})}{P(D_{positive}|H_{diseased})P(H_{diseased}) + P(D_{positive}|H_{healthy})P(H_{healthy})} \\ =& \frac{1\% * 99\%}{99\% * 1\% + 1\% * 99\%} \\ =& 0.5 \end{aligned} \tag{16}$$

Although the accuracy of this test is 99% which is pretty high, there are only 50% chance to have this disease if you get positive from this test because the percentage of this disease is just 1%. Hence, comparing with a small amount of patients, even 1% healthy people is sizable.

7 AP-RC

Theorem 7.1. $0 \leq K(x|\ell(x)) - O(1) \leq -\log_2 M(x) \leq Km(x) \leq K(x) \leq \ell(x) + 2\log_2 \ell(x) + O(1)$

We'll prove each of the five inequalities and combine the results.

Lemma 7.1.1. $0 \leq K(x|\ell(x)) - O(1)$

Proof. By definition,

$$K(x|\ell(x)) := \min_p \{\ell(p) : U(\ell(x)'p) = x\}.$$

But as $\forall p, \ell(p) > 0$, we have $K(x|\ell(x)) \geq 0$. For any negative constant c, $c \leq 0 \leq K(x|\ell(x))$. We've shown that

$$0 \leq K(x|\ell(x)) - c$$

for some $O(1)$ term c.

□

Lemma 7.1.2. $K(x|\ell(n)) - O(1) \leq -\log_2 M(x)$.

Proof. By definition of M,

$$\sum_{x \in \mathcal{X}^n} M(x) = \sum_{x \in \mathcal{X}^n} \sum_{p:U(p)=x*} 2^{-\ell(p)}.$$

Let $x, y \in \mathcal{X}^n$, if $x \neq y$, then $x* \neq y*$ as one cannot be prefix of the other. Hence $\forall p, q$ such that $U(p) = x*$ and $U(q) = y*$, we must have $p \neq q$. We then get

$$\sum_{x \in \mathcal{X}^n} \sum_{p:U(p)=x*} 2^{-\ell(p)} = \sum_{p:\exists x \in \mathcal{X}^n, U(p)=x*} 2^{-\ell(p)}.$$

Note that $A = \{p : \exists x \in \mathcal{X}^n, U(p) = x*\}$ forms a prefix code, since if there exist $p, q \in A$ and $p = qr$ for some $r \neq \epsilon$. Then $U(q) = x*$ for some $x \in \mathcal{X}^n$, and $U(qs)$ output a string starting with x, so we cannot have $U(qs) = y*$ for $y \in \mathcal{X}^n$ and $y \neq x$. But since $q \in A$, we can only have $U(p) = x*$, which contradicts the fact that for a given x,p:U(p)=x* forms a prefix code.
We now have

$$\sum_{x \in \mathcal{X}^n} M(x) = \sum_{p:\exists x \in \mathcal{X}^n, U(p)=x*} 2^{-\ell(p)} \leq 1$$

by Kraft's inequality.
We also know from the lecture that M is enumerable.

Let $s_x = \lceil -\log_2 M(x) \rceil \in \mathbb{N}$. We have

$$\sum_{x \in \mathcal{X}^n} 2^{-s_x} \leq \sum_{x \in \mathcal{X}^n} M(x) \leq 1.$$

$\Rightarrow$: $\exists$ prefix code p for $x \in \mathcal{X}^n$ with $\ell(p) = s_x$ (by Kraft inequality).
Since the proof for Kraft inequality for known $\sum_{x \in \mathcal{X}^n} M(x)$ is constructive, there exists an effective prefix code in the sense that $\exists$prefix Turing machine $T_n : \forall x \in \mathcal{X}^\backslash, \exists p : T_n(p) = x$, and $\ell(p) = s_x$.
Theorem 2.9 gives $K(x) \overset{+}{<} K_T(x) + K_U(T)$. Conditioned to $n = \ell(x)$, we have

$$K(x|n) \overset{+}{<} K_T(x|n) + K_U(T|n).$$

When n is known, $K_U(T_n)$ is a constant. Hence, $K_U(T_n|n) = O(1)$ and $K_{T_n}(x|n) \leq K_{T_n}(x) \leq s_x$. We conclude that

$$K(x|n) \overset{+}{<} K_T(x|n) + K_U(T|n) \leq s_x + O(1) \leq -\log M(x) + O(1),$$

which is equivalent to the second inequality we want to prove:

$$K(x|\ell(n)) - O(1) \leq -\log_2 M(x).$$

□

Lemma 7.1.3. $KM(x) \leq Km(x)$

Proof. By the definition of monotone Kolmogorov complexity, $\exists p' : \ell(p') = Km(x)$ and $U(p') = x*$. Hence,

$$\begin{aligned} M(x) &= \sum_{p:U(p)=x*} 2^{-\ell(p)} \\ &= 2^{-\ell(p')} + \sum_{\substack{p:U(p)=x* \\ \text{and } p\neq p'}} 2^{-\ell(p)} \\ &\geq 2^{-\ell(p')} \\ &= 2^{-Km(x)}. \end{aligned}$$

Therefore, $\log_2 M(x) \leq -Km(x)$.
$\Rightarrow -\log_2 M(x) \leq Km(x)$ □

Lemma 7.1.4. $Km(x) \leq K(x)$

Proof. By the definition of the prefix complexity, $\exists p : \ell(p) = K(x)$ and $U(p) = x$. When U halts, exactly p is to the left of input head and x is to the left of the output head. Denote $p = p_1p_2...p_n$, where $p_1, p_2, ..., p_n$ are digits of p. When the last bit of x is output, to the left of the input head must be $p_1p_2...p_i$ for some $i \leq n$, since both the input type and output type are unidirectional. By the definition of the monotone complexity,

$$Km(x) \leq i \leq n = \ell(p) = K(x)$$

□

Lemma 7.1.5. $K(x) \leq \ell(x) + 2\log_2 \ell(x) + O(1)$

Proof. It was shown in the lecture that $\ell(x\text{`}) = \ell(x) + 2\log_x \ell(x) + O(1)$. As $x$‘ is the second order prefix coding, $\exists$ Turing machine $T : T(x\text{`}) = x$ and $K(T) = O(1)$.
By construction, $K_T(x) \leq \ell(x\text{`}) = \ell(x) + 2\log_x \ell(x) + O(1)$. Therefore,

$$K(x) \leq K_T(x) + K(T) = K_T(x) + O(1) \leq \ell(x) + 2\log_x \ell(x) + O(1).$$

□

Combining the five inequalities, we've proved the Theorem 7.1.

8 AP-CM

(i) Let $x_n = E[(M(0|x_{<n}) - \mu(0|x_{<n}))^2]$.
Since $(M(0|x_{<n}) - \mu(0|x_{<n}))^2 \geq 0$, we have $x_n \geq 0, \forall n \in \mathbb{N}$.
Let $s_n = \sum_{t=1}^{n} x_t$.
Since $s_{n+1} - s_n = x_{n+1} \geq 0, \forall n \in \mathbb{N}$, $(s_n) = (s_1, s_2, s_3, ...)$ is a monotonic increasing sequence.
We know from the Solomonoff's bound that

$$\sum_{t=1}^{\infty} \sum_{x_{<t} \in \mathbb{B}^{t-1}} \mu(x_{<t})(M(0|x_{<t}) - \mu(0|x_{<t}))^2 \stackrel{+}{<} \frac{1}{2} \ln 2K(\mu) < \infty.$$

By our definition,

$$s_\infty = \sum_{t=1}^{\infty} E[(M(0|x_{<t}) - \mu(0|x_{<t}))^2] = \sum_{t=1}^{\infty} \sum_{x_{<t} \in \mathbb{B}^{t-1}} \mu(x_{<t})(M(0|x_{<t}) - \mu(0|x_{<t}))^2.$$

Hence, $s_n < s_\infty \stackrel{+}{<} \frac{1}{2} \ln 2K(\mu) < \infty$. (s_n) is a bounded sequence. By the monotone convergence theorem, a bounded monotonic increasing sequence converges and the limit is its supremum. Therefore, (s_n) converges to some $U \in \mathbb{R}$.
From the definition of convergence, $\forall \epsilon, \exists N$, such that $\forall n > N, |s_n - U| < \epsilon$, which implies

$$x_{n+1} = s_{n+1} - s_n \leq U - s_n < \epsilon$$

(where we used the fact that $U - s_n = s_\infty - s_n = \sum_{t=n+1}^{\infty} x_t \geq x_{n+1} = s_{n+1} - s_n$).
Hence $\forall \epsilon, \exists N$, such that $\forall n > N, s_n < \epsilon$. The sequence (x_n) converges to 0.
By applying Markov inequality, we have

$$0 \leq P[(M(0|x_{<t}) - \mu(0|x_{<t}))^2 \geq \epsilon] \leq \frac{x_t}{\epsilon}.$$

Since $\forall \epsilon > 0$,

$$\lim_{t \to \infty} \frac{x_t}{\epsilon} = \frac{1}{\epsilon} \lim_{t \to \infty} x_t = 0.$$

It follows from the comparison test (for the sequences) that

$$\lim_{t \to \infty} P[(M(0|x_{<t}) - \mu(0|x_{<t}))^2 \geq \epsilon] = 0.$$

We have

$$\lim_{t \to \infty} P[(M(0|x_{<t}) - \mu(0|x_{<t}))^2 < \epsilon] = 1.$$

$(M(0|x_{<t}) - \mu(0|x_{<t}))^2 < \epsilon$ iff $|M(0|x_{<t}) - \mu(0|x_{<t})| < \sqrt{\epsilon}$. Since ϵ is arbitrary, we can say that $M(0|x_{<t}) - \mu(0|x_{<t}$ tends to zero as $t \to \infty$ with μ-probability 1.

(ii) By the Solomonoff's bound, we have

$$\sum_{t=1}^{\infty} E[(M(0|x_{<t}) - \mu(0|x_{<t}))^2] = \sum_{t=1}^{\infty} \sum_{x_{<t} \in \mathbb{B}^{t-1}} \mu(x_{<t})(M(0|x_{<t}) - \mu(0|x_{<t}))^2 < \frac{1}{2} \ln 2K(\mu) + O(1).$$

Hence, using Karkov inequality, we get

$$\begin{aligned}
&\sum_{t=1}^{\infty} P[(M(0|x_{<t}) - \mu(0|x_{<t}))^2 \geq \epsilon^2] \\
\leq &\sum_{t=1}^{\infty} \frac{E[(M(0|x_{<t}) - \mu(0|x_{<t}))^2]}{\epsilon^2} \\
= &\frac{1}{\epsilon^2} \sum_{t=1}^{\infty} E[(M(0|x_{<t}) - \mu(0|x_{<t}))^2] \\
< &\frac{\ln 2K(\mu)}{2\epsilon^2} + O(1) \\
= &\frac{\ln 2K(\mu) + O(1)}{2\epsilon^2} \\
< &\frac{\ln 2K(\mu) + O(1)}{\epsilon^2} \\
= &\frac{c}{\epsilon^2},
\end{aligned} \tag{17}$$

where $c = \ln 2K(\mu) + O(1)$. Let $I_{|M(0|x_{<t})-\mu(0|x_{<t})|>\epsilon}$ be an indicator function (i.e. $I_{|M(0|x_{<t})-\mu(0|x_{<t})|>\epsilon} = 1$ if $|M(0|x_{<t}) - \mu(0|x_{<t})| > \epsilon, 0$ otherwise). The expected number of times t in which $|M(0|x_{<t}) - \mu(0|x_{<t})| > \epsilon$ is

$$\begin{aligned}
E(\sum_{t=1}^{\infty} I_{|M(0|x_{<t})-\mu(0|x_{<t})|>\epsilon}) &= \sum_{t=1}^{\infty} E(I_{|M(0|x_{<t})-\mu(0|x_{<t})|>\epsilon}) \\
&= \sum_{t=1}^{\infty} P[|M(0|x_{<t}) - \mu(0|x_{<t}))| \geq \epsilon] \\
&= \sum_{t=1}^{\infty} P[(M(0|x_{<t}) - \mu(0|x_{<t}))^2 \geq \epsilon^2].
\end{aligned} \tag{18}$$

Combining the results of 18 and 17, we see that the expected number of times t in which $|M(0|x_{<t}) - \mu(0|x_{<t})| > \epsilon$ is finite and bounded by $\frac{c}{\epsilon^2}$.

Let $X = \sum_{t=1}^{\infty} I_{|M(0|x_{<t})-\mu(0|x_{<t})|>\epsilon}$ be the number of ϵ deviations. It is shown that $E(X) < \frac{c}{\epsilon^2}$. Therefore, applying Markov inequality, we have

$$P(X > \frac{c}{\epsilon^2 \delta}) < \frac{E(X)}{(\frac{c}{\epsilon^2 \delta})} = \delta \frac{E(X)}{c/\epsilon^2} < \delta.$$

(iii)

Theorem 8.1. $M(1|0^n) \overset{\times}{=} 2^{-K(n)}$

To prove the theorem, we will need the following lemmas.

Lemma 8.1.1. $\frac{1}{M(0^n)} = O(1)$

Proof. $\exists$ Turing machine $T_i : T_i$ prints 0's consecutively given any input.
Let the universal Turing machine U simulates T_i. Let $c = \epsilon$‘i‘ϵ, $\forall n \in \mathbb{N}, U(c) = 0^n*$, since $\forall n$, c is to the left of the input head when the last bit of 0^n is output.
Hence $Km(0^n) \leq \ell(c)$, which implies $2^{-Km(0^n)} \geq 2^{-\ell(c)}$.

As $M(x) \geq 2^{-Km(x)}, \forall x$, $M(0^n) \geq 2^{-\ell(c)}$.
We also see that $2^{\ell(c)} = O(1) > 0$ as a description of T_i does not depend on n. Combining our results, we have

$$0 < \frac{1}{M(0^n)} \leq \frac{1}{2^{-\ell(c)}} = 2^{\ell(c)} = O(1).$$

$\frac{1}{M(0^n)}$ is bounded by constants. Hence, $\frac{1}{M(0^n)} = O(1)$. □

Lemma 8.1.2. $M(0^n1) \overset{\times}{>} 2^{-K(n)}$

Proof. By the definition of prefix complexity, $\exists p = y\text{`}i\text{`}q : \ell(p) = K(n), U(p) = n$.
We can construct a prefix Turing machine T such that $T(n) = 0^n1, \forall n \in \mathbb{N}$. $T = T_j$ for some j as the set of Turing machines is countable.
Consider a universal Turing machine U'. Given input $p\text{`}j\text{`}r$, where $p = y\text{`}i\text{`}q$, and r is a short compiler which simulates T_i and T_j on U. U' first simulate T_i on $y\text{`}q$. It gets $z = T_i(y\text{`}q) = U(y\text{`}i\text{`}q) = U(p)$ for some z, and then simulates T_j on z and output $T_j(z)$.
Since $U(p) = n$ and $T_j(n) = 0^n1$, $U'(p\text{`}j\text{`}r) = T_j(U(p)) = 0^n1$. Hence $K_{U'}(0^n1) \leq \ell(p\text{`}j\text{`}r)$.
We know $\ell(j) = O(1)$ as the description of T_j does not depend on n, and $\ell(r) = O(1)$ by the short compiler assumption. We can conclude that

$$K_{U'}(0^n1) \leq \ell(p\text{`}j\text{`}r) \leq \ell(p) + \ell(j) + \ell(r) + O(1) = \ell(p) + O(1) = K(n) + O(1). \tag{19}$$

Following Theorem 2.9 and 2.20 from the slides, we have

$$Km(0^n1) \leq \ell(p\text{`}j\text{`}r) \leq K(0^n1) \leq K_{U'}(0^n1). \tag{20}$$

Combining 19 and 20, we have $Km(0^n1) \leq K(n) + O(1)$. It follows that

$$-Km(0^n1) \geq -K(n) - O(1)$$

$$\Rightarrow 2^{-Km(0^n1)} \geq 2^{-K(n)-O(1)} = 2^{-K(n)}2^{-O(1)}$$

$$\Rightarrow 2^{-Km(0^n1)} \geq O(1)2^{-K(n)}$$

$$\Rightarrow 2^{-K(n)} = O(2^{-Km(0^n1)}).$$

As $2^{-Km(0^n1)} \leq M(0^n1)$ by Theorem 4.2, we have $2^{-K(n)} = O(M(0^n1))$, which is equivalent to $M(0^n1) \overset{\times}{>} 2^{-K(n)}$. □

Lemma 8.1.3. $M(0^n1) \overset{\times}{<} 2^{-K(n)}$

Proof. Let $P(n) = M(0^n1)$.

$$\sum_{n\in\mathbb{N}} P(n) = \sum_{n\in\mathbb{N}} M(0^n1) = \sum_{n\in\mathbb{N}} \sum_{U(p)=0^n1*} 2^{-\ell(p)}.$$

Let $x = 0^k1, y = 0^j1$ for some $j, k \in \mathbb{N}, j \neq k$, then $x* \neq y*$ as one cannot be prefix of the

other. Hence $\forall p, q$ such that $U(p) = x*$ and $U(q) = y*$, we must have $p \neq q$. We then get

$$\sum_{n\in\mathbb{N}} \sum_{U(p)=0^n1*} 2^{-\ell(p)} = \sum_{p:\exists n\in\mathbb{N}, U(p)=0^n1*} 2^{-\ell(p)}.$$

Let $A = \{p : \exists n \in \mathbb{N}, U(p) = 0^n1*\}$. We claim that A forms a prefix code. Suppose there exist $p, q \in A$ and $p = qr$ for some $r \neq \epsilon$. Then $U(q) = 0^j1*$ for some $j \in \mathbb{N}$, and $U(qs)$ output a string starting with 0^j1, so we cannot have $U(qs) = 0^k1*$ for $k \in \mathbb{N}$ and $k \neq j$. But since $q \in A$, we can only have $U(p) = 0^j1*$, which contradicts the fact that for a given x, $\{p : U(p) = x*\}$ forms a prefix code.
We now have

$$\sum_{n\in\mathbb{N}} P(n) = \sum_{p\in A} 2^{-\ell(p)} \leq 1$$

by Kraft's inequality.
It follows from Theorem 2.17 on the slides that

$$K(n) < -\log P(n) + K(P) + O(1).$$

Obviously $K(M)$ does not depend on n. Hence, $K(P) = O(1)$. We get

$$K(n) < -\log P(n) + O(1) \Rightarrow -K(n) > \log M(0^n1) + O(1)$$

$$\Rightarrow 2^{-K(n)} > O(1)M(0^n1)$$

$$\Rightarrow M(0^n1) \overset{\times}{<} 2^{-K(n)}$$

□

By definition,

$$M(1|0^n) = \frac{M(0^n1)}{M(0^n)} = \frac{1}{M(0^n)}M(0^n1).$$

Since $\frac{1}{M(0^n)} = O(1)$ by Lemma 8.1.1,

$$M(1|0^n) = O(1)M(0^n1),$$

which is equivalent to $M(1|0^n) \overset{\times}{=} M(0^n1)$.
Combining Lemma 8.1.2 and 8.1.3, we have $M(0^n1) \overset{\times}{=} 2^{-K(n)}$.
It can be easily shown that the relation $\overset{\times}{=}$ is transitive: if $f \overset{\times}{=} g$ and $g \overset{\times}{=} h$, we have $f \overset{\times}{<} g, f \overset{\times}{>} g, g \overset{\times}{<} h$ and $g \overset{\times}{>} h$. $f = O(g)$ and $g = O(h)$ implies $f = O(h)$, and similarly $h = O(f)$. Therefore, $f \overset{\times}{=} h$.
Combining our results,

$$M(1|0^n) \overset{\times}{=} M(0^n1) \wedge M(0^n1) \overset{\times}{=} 2^{-K(n)}$$

$$\Rightarrow M(1|0^n) \overset{\times}{=} 2^{-K(n)}$$

Implications for sequential predictions We've shown that $\lim_{n\to\infty} K(n) = \infty$. Therefore, $\lim_{n\to\infty} 2^{-K(n)} = 0$. Combining this with the result we have here: $M(1|0^n) \overset{\times}{=} 2^{-K(n)}$, we notice that $M(1|0^n)$ goes to 0 as n goes to ∞. This agrees with our intuition that when a sequence starts with a very long binary string full of 0's, the next digit is very likely to be 0. When n is some 'simple' number, such as 10000 or 1111, $K(n)$ might be smaller compared with neighbouring numbers, and we're confident that the next digit will be 0.

(iv) Let p be a program that $U(p) = x*$. Since p is the minimal program, U does not output $x*$ given any prefix of p.
Let Γ_p denote set of (infinite) sequences starts with p. $\forall w \in \Gamma_p$, U outputs $x*$ when w is the input.
Since the input tape of U is binary with no black symbols, we treat the input as a binary sequence. Let w denote the input,

$$P(w \in \Gamma_p) = (\frac{1}{2})^{\ell(p)} = 2^{-\ell(p)}.$$

Let $A = p : T(p) = x*$. We know (from Definition 2.6) that A forms a prefix code. Hence, $\forall q, r \in A, q \neq r$ implies $\Gamma_q \cap \Gamma_r = \emptyset$.
Since for any w such that U output $x*$ given w, we can find a prefix p of w such that p is to the left of the input head when the last bit of x is output, we conclude that $\forall w \in \mathbb{B}^\infty$, if U outputs $x*$ given w as input, $\exists p : w \in \Gamma_p$ and $U(p) = x*$.
Let w be a binary sequence of uniform random noise,

$$\begin{aligned} P(\text{T prints } x* \text{ given } w) =& P(w \in \cup_{p\in A}\Gamma_p) \\ =& \sum_{p\in A} P(\Gamma_p) \text{ (as the sets are disjoint)} \\ =& \sum_{p:T(p)=x*} 2^{-\ell(p)} \end{aligned} \tag{21}$$

9 MDL-ML

By assumption , $y|x_i \sim \mathcal{N}(f_d(x_i), \sigma^2)$.
We write $y_i = f_d(x_i) + \epsilon_i$, where $\epsilon_1, ..., \epsilon_n \sim$i.i.d. $\mathcal{N}(0, \sigma^2)$ (n is the number of data points).
Let $\mathbf{X}$ be a n by $d+1$ matrix, and denote x_{ij} as the element on the i^{th} row, j^{th} column of $\mathbf{X}$. We can define $\mathbf{X}$ as $x_{ij} = x_i^{j-1}, \forall i = 1, ..., n, \forall j = 1, ..., d+1$.
Now let $\mathbf{a_{0:d}}$ be the $d+1$-dimensional vector representing the coefficients:

$$\mathbf{a_{0:d}} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix}, \text{ and let } \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}.$$

We see that

$$\mathbf{X}\mathbf{a_{0:d}} = \begin{bmatrix} f_d(x_1) \\ \vdots \\ f_d(x_n) \end{bmatrix},$$

since the i^{th} element in $\mathbf{X}\mathbf{a_{0:d}}$ is the dot product of the i^{th} row of $\mathbf{X}$ and $\mathbf{a_{0:d}}$:

$$\text{row}_i(\mathbf{X})\mathbf{a_{0:d}} = \begin{bmatrix} 1 & x_i & x_i^2 & \dots & x_i^d \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix} = f_d(x_i).$$

We now have a formula for the sum of squares error using the notations just defined:

$$\begin{aligned} \text{SQ}(\mathbf{a_{0:d}}) &= \sum_{i=1}^{n} (y_i - f_d(x_i))^2 \\ &= \sum_{i=1}^{n} \epsilon_i^2 \\ &= \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \\ &= (\mathbf{y} - \mathbf{X}\mathbf{a_{0:d}})^T (\mathbf{y} - \mathbf{X}\mathbf{a_{0:d}}) \\ &= (\mathbf{y}^T - \mathbf{a_{0:d}}^T \mathbf{X}^T)(\mathbf{y} - \mathbf{X}\mathbf{a_{0:d}}) \\ &= \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{a_{0:d}} - \mathbf{a_{0:d}}^T\mathbf{X}^T\mathbf{y} + \mathbf{a_{0:d}}^T\mathbf{X}^T\mathbf{X}\mathbf{a_{0:d}} \\ &= \mathbf{y}^T\mathbf{y} - 2\mathbf{a_{0:d}}^T\mathbf{X}^T\mathbf{y} + \mathbf{a_{0:d}}^T\mathbf{X}^T\mathbf{X}\mathbf{a_{0:d}} \end{aligned} \tag{22}$$

The last equality holds as $\mathbf{y}^T\mathbf{X}\mathbf{a_{0:d}}$ is a scalar. Hence $\mathbf{y}^T\mathbf{X}\mathbf{a_{0:d}} = (\mathbf{y}^T\mathbf{X}\mathbf{a_{0:d}})^T = \mathbf{a_{0:d}}^T\mathbf{X}^T\mathbf{y}$. Take the derivative with respect to $\mathbf{a_{0:d}}$,

$$\frac{d}{d\mathbf{a_{0:d}}}\text{SQ}(\mathbf{a_{0:d}}) = \frac{d}{d\mathbf{a_{0:d}}}(\mathbf{y}^T\mathbf{y} - 2\mathbf{a_{0:d}}^T\mathbf{X}^T\mathbf{y} + \mathbf{a_{0:d}}^T\mathbf{X}^T\mathbf{X}\mathbf{a_{0:d}}) = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\mathbf{a_{0:d}}.$$

Since we know from the lecture that

$$\mathbf{a_{0:d}^{ML}} = \arg\min_{\mathbf{a_{0:d}}} \text{SQ}(\mathbf{a_{0:d}}),$$

we just need to set $\frac{d}{d\mathbf{a_{0:d}}}\text{SQ}(\mathbf{a_{0:d}})$ to 0 and solve:

$$\begin{aligned} -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\mathbf{a_{0:d}^{ML}} &= 0 \\ \iff \mathbf{X}^T\mathbf{X}\mathbf{a_{0:d}^{ML}} &= \mathbf{X}^T\mathbf{y} \\ \iff \mathbf{a_{0:d}^{ML}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \end{aligned}$$

$\mathbf{a_{0:d}^{ML}}$ exists and is unique as long as $\mathbf{X}^T\mathbf{X}$ is invertible.

10 MDL-Ber

(i) Let $l(\theta) = \ln P(x|\theta)$ be the log-likelihood function.

$$\hat{\theta} := \arg\max_{\theta} \ln P(x|\theta) = \arg\max_{\theta} l(\theta)$$

$$l(\theta) = k \ln \theta + (n-k) \ln(1-\theta)$$

$$\Rightarrow l'(\theta) = \frac{k}{\theta} - \frac{n-k}{1-\theta}$$

We find that the second derivative is negative:

$$l''(\theta) = \frac{k}{\theta^2} - \frac{n-k}{(1-\theta)^2} < 0.$$

It follows that $l(\theta)$ reaches maximum when $l'(\theta) = 0$. Setting $l'(\hat{\theta}) = 0$ and we have $\hat{\theta} = \frac{k}{n}$

(ii) Since n and k are natural numbers, n'k' (the second order prefix coding) is a code for $\hat{\theta}$.

$$K(\hat{\theta}) \overset{+}{<} K(n) + K(k) \overset{+}{<} \log n + 2 \log\log n + \log k + 2 \log\log k \leq 2 \log n + 4 \log\log n$$

(iii)

$$Km(x) \leq -\log_2 \mu(x) + K(\mu),$$

where $\mu(x) := P(x|\hat{\theta})$.

(iv) Suppose $\langle\tilde{\theta}\rangle = x_1 x_2 ... x_m$ (m bits), and $\tilde{\theta} = \sum_{i=1}^{m} x_i (\frac{1}{2})^i$.
We may have $|\tilde{\theta} - \hat{\theta}| < \frac{1}{2^m}$. Set $\frac{1}{2^m} = n^{-1/2}$, we get $m = \frac{1}{2} \log_2 n$.
We need $\lceil \frac{1}{2} \log_2 n \rceil$ bits to get accuracy $|\tilde{\theta} - \hat{\theta}| < n^{-1/2}$.

(v) Since $\hat{\theta}$ is the MLE, $P(x|\hat{\theta}) \geq P(x|\theta), \forall\theta$.
Therefore, $P(x|\hat{\theta}) \geq P(x|\tilde{\theta})$

$$\Rightarrow \frac{P(x|\hat{\theta})}{P(x|\tilde{\theta})} \geq 1$$

$$\Rightarrow \log P(x|\hat{\theta}) - \log P(x|\tilde{\theta}) = \log \frac{P(x|\hat{\theta})}{P(x|\tilde{\theta})} \geq \log 1 = 0$$

To show the second inequality, note that

$$\begin{aligned}
\log \frac{P(x|\hat{\theta})}{P(x|\tilde{\theta})} &= \log \frac{\hat{\theta}^k (1-\hat{\theta})^{n-k}}{\tilde{\theta}^k (1-\tilde{\theta})^{n-k}} \\
&= k \log \frac{\hat{\theta}}{\tilde{\theta}} + (n-k) \log \frac{1-\hat{\theta}}{1-\tilde{\theta}} \\
&= n[\frac{k}{n} \log \frac{\hat{\theta}}{\tilde{\theta}} + (1 - \frac{k}{n} \log \frac{1-\hat{\theta}}{1-\tilde{\theta}})] \\
&= n[\hat{\theta} \log \frac{\hat{\theta}}{\tilde{\theta}} + (1 - \hat{\theta} \log \frac{1-\hat{\theta}}{1-\tilde{\theta}})] \\
&= nKL(\hat{\theta}||\tilde{\theta}) \\
&\approx nc(\hat{\theta})(\hat{\theta} - \tilde{\theta})^2.
\end{aligned}$$

As $|\hat{\theta} - \tilde{\theta}| < (n^{-1/2})^2 = \frac{1}{n}$,

$$nc(\hat{\theta})(\hat{\theta} - \tilde{\theta})^2 < c(\hat{\theta})n\frac{1}{n} = c(\hat{\theta}) = O(1).$$

We can conclude that

$$\log P(x|\hat{\theta}) - \log P(x|\tilde{\theta}) = \log \frac{P(x|\hat{\theta})}{P(x|\tilde{\theta})} \leq O(1).$$

(vi) Let $q(x) = P(x|\tilde{\theta})$. By Theorem 2.20,

$$Km(x) \leq -\log_2 q(x) + K(q) + O(1).$$

$K(q)$ only depends on $\tilde{\theta}$ since we can have a Turing machine which compute $P(x|\theta)$ given input x and θ. Hence, $K(q) = K(\tilde{\theta}) + O(1)$.
We've shown that $\lceil \frac{1}{2} \log_2 n \rceil$ bits is enough for describing $\tilde{\theta}$. Then, we have

$$K(q) \leq \lceil \frac{1}{2} \log_2 n \rceil + O(1) = \frac{1}{2} \log_2 n + O(1).$$

We've also shown in the previous question that $P(x|\tilde{\theta}) \leq P(x|\hat{\theta})$.
Hence we can combine our results:

$$Km(x) \leq -\log_2 P(x|\hat{\theta}) + \frac{1}{2} \log_2 n + O(1).$$

11 BSP-PO

From Definition 7.1, $\xi_U(x_t \mid x_{<t}) = \sum_\nu \omega_\nu \nu(x_t \mid x_{<t})$ with $\nu \in \mathcal{M}_U$
For total square loss, If there exists a ρ that $\mathcal{F}(\nu, \rho) \leq \mathcal{F}(\nu, \xi_U)$. This implies

$$\begin{aligned}
0 &\leq \mathcal{F}(\nu, \xi_U) - \mathcal{F}(\nu, \rho) \\
&= \sum_\nu \omega_\nu \sum_{x_t} \Big[(\nu(x_t \mid x_{<t}) - \xi_U(x_t \mid x_{<t}))^2 - (\nu(x_t \mid x_{<t}) - \rho(x_t \mid x_{<t}))^2 \Big] \\
&= \sum_\nu \omega_\nu \sum_{x_t} \Big[\nu(x_t \mid x_{<t})^2 - 2\nu(x_t \mid x_{<t})\xi_U(x_t \mid x_{<t}) + \xi_U(x_t \mid x_{<t})^2 \\
&- \nu(x_t \mid x_{<t})^2 + 2\nu(x_t \mid x_{<t})\rho(x_t \mid x_{<t}) - \rho(x_t \mid x_{<t})^2 \Big] \\
&= \sum_\nu \omega_\nu \sum_{x_t} \Big[-2\nu(x_t \mid x_{<t})\xi_U(x_t \mid x_{<t}) + \xi_U(x_t \mid x_{<t})^2 \\
&+ 2\nu(x_t \mid x_{<t})\rho(x_t \mid x_{<t}) - \rho(x_t \mid x_{<t})^2 \Big] \\
&= \sum_{x_t} \Big[-2\xi_U(x_t \mid x_{<t})^2 + \xi_U(x_t \mid x_{<t})^2 + 2\xi_U(x_t \mid x_{<t})\rho(x_t \mid x_{<t}) - \rho(x_t \mid x_{<t})^2 \Big] \\
&= \sum_{x_t} -\Big[\xi_U(x_t \mid x_{<t}) - \rho(x_t \mid x_{<t}) \Big]^2 \\
&\leq 0
\end{aligned}$$

Hence, $\xi_U(x_t \mid x_{<t}) - \rho(x_t \mid x_{<t}) = 0 \forall x_t \implies \xi_U = \rho$
and $\mathcal{F}(\nu, \xi) = \mathcal{F}(\nu, \rho)$
Since the only ρ satisfying $\mathcal{F}(\nu, \rho) \leq \mathcal{F}(\nu, \xi_U)$ is $\rho = \xi_U$, but there is no strict inequality when this happens. $\Rightarrow \xi$ is pareto-optimal $w.r.t$ total square loss
Similarly, for total KL loss, if there exists a ρ that $\mathcal{F}(\nu, \rho) \leq \mathcal{F}(\nu, \xi_U)$. This implies

$$\begin{aligned}
0 &\leq \mathcal{F}(\nu, \xi_U) - \mathcal{F}(\nu, \rho) \\
&= \sum_{\nu} \omega_\nu \sum_{x_t} \Big[\nu(x_t \mid x_{<t}) \log \frac{\nu(x_t \mid x_{<t})}{\xi(x_t \mid x_{<t})} - \nu(x_t \mid x_{<t}) \log \frac{\nu(x_t \mid x_{<t})}{\rho(x_t \mid x_{<t})} \Big] \\
&= \sum_{\nu} \omega_\nu \sum_{x_t} \nu(x_t \mid x_{<t}) \log \frac{\rho(x_t \mid x_{<t})}{\xi(x_t \mid x_{<t})} \\
&= \sum_{x_t} \xi(x_t \mid x_{<t}) \log \frac{\rho(x_t \mid x_{<t})}{\xi(x_t \mid x_{<t})} \\
&\leq 0
\end{aligned}$$

Hence, $\mathcal{F}(\nu, \xi) = \mathcal{F}(\nu, \rho)$
$\Rightarrow \xi = \rho$
$\Rightarrow \xi$ is pareto-optimal $w.r.t$ total KL loss

12 USM-NCD

1. Installation

   Computer system is macOS Sierra(v 10.12.6). Complearn depends on many libraries and not friendly with macOS, hence Docker is a better choose.

   - Build Dockerfile as follow:

     ```
     FROM debian/eol:lenny
     RUN apt-get update
     RUN apt-get install -y complearn-tools libcomplearn-dev
     qsearch-tools libqsearch-dev zlib1g-dev graphviz
     ADD examples /tmp/
     ```

     (Note: grahviz is a graph library. and the folder **examples** contains **34-mammals** and **73-languages**, which we need to use in this assignment.)

   - Use this dockerfile to build and then run this container.

     ```
     docker build -t complearn .
     docker run -i -t complearn
     ```

2. Languages Data

   In the initial example, there are mammals example but no language example. Therefore, we need to grab some language data online. At the first, I thought about Wikipedia. Build a spyder to grab the pages with same keyword but different languages. In this case, I got the page of "Artificial intelligence" with 64 languages. However, I realized that this method might get

huge bias, because even they are same topic but still have wide difference in content. Finially, I use the same source as slide 6. Build a spyder to grab 73 kinds of language translation of this same document, which you can find in https://gitlab.cecs.anu.edu.au/u5976992/Advance-AI-ass1/tree/master/73-languages.

3. Command Line

   We should familiar with 3 command line, **ncd**, **maketree**, **neato** and use them to do this assignment.

   - **ncd**

     NCD (Normalized Compression Distance), is the key idea of this algorithm, which we already learned in slides6. For finish this job, we should know same options:

     ```
     -b enable binary output mode for matrix
     -o output file name
     -c set compressor to use, such as bzlib, zlib, blocksort
     -d directory-mode, which allows us to do with a directory of files
     ```

     For this assignment, we will do **ncd -b -o mammal -d /34mammals /34mammals**(for languages tree, change the output file name and dictionary)

   - **maketree**

     We can use it to create an unrooted binary tree by distance matrix.

     maketree mammal.clb(for languages tree, change to language.clb)

   - **neato**

     This command will create a visual representation of our tree in postscript format, such png, jpg.

     neato -Tpng treefile.dot > 34mammals-unrooted.png

4. Result

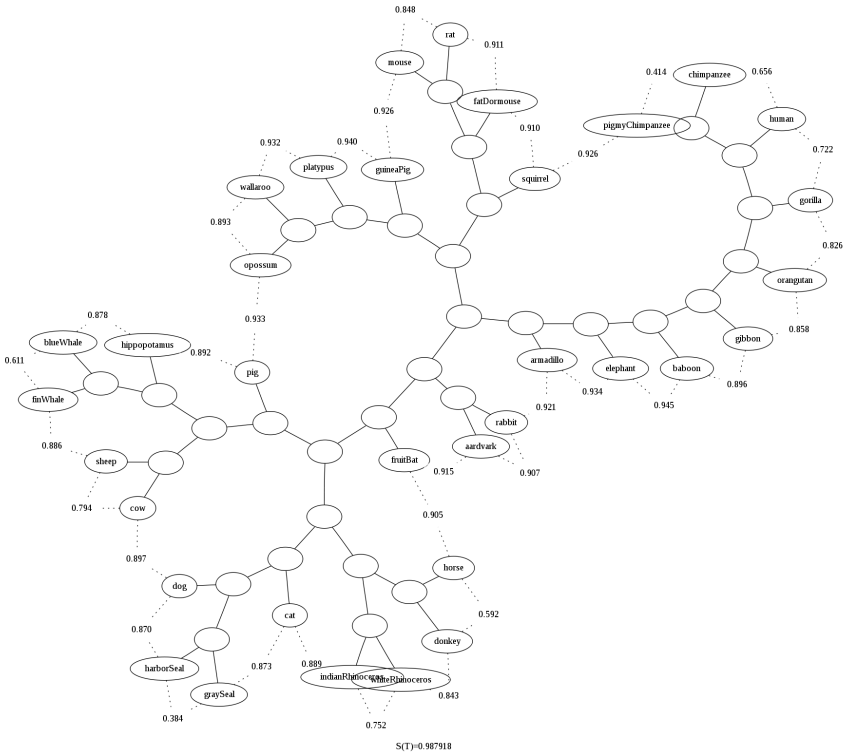

Figure 1: mammals

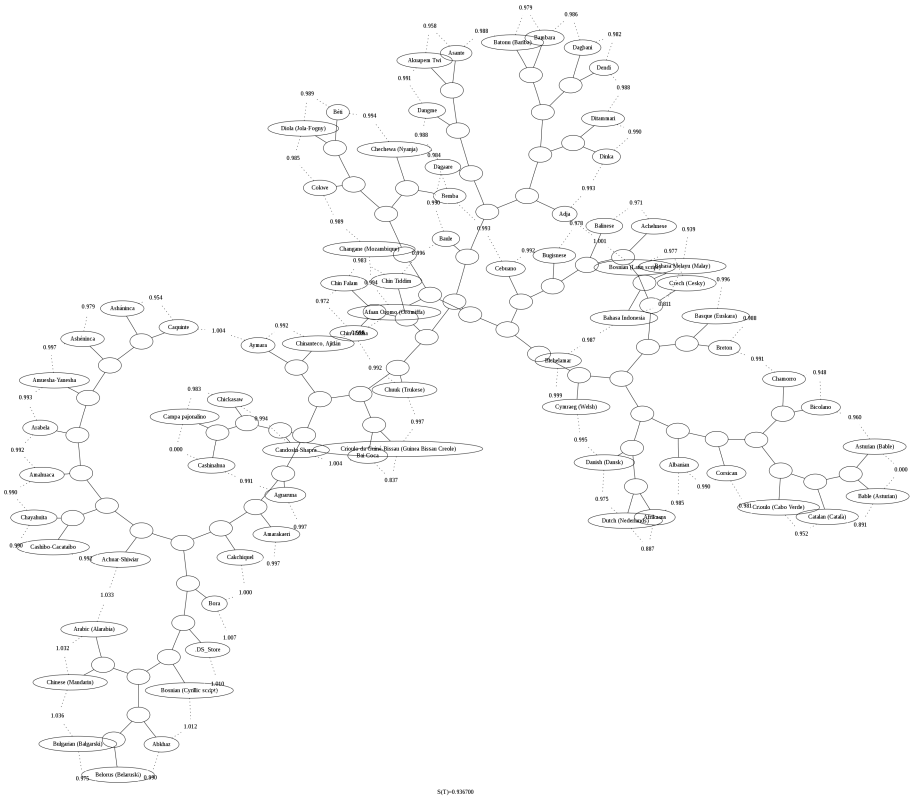

Figure 2: languages

References

[1] Marcus Hutter *Universal Artificial Intelligence* Springer, Berlin, 2005

[2] Ming Li, Paul Vitányi *An Introduction to Kolmogorov Complexity and Its Applications* Springer-Verlag, New York, 1993

[3] Peter D. Hoff. *A First Course in Bayesian Statistical Methods.* University of Washington, Seattle, 2010