Kolmogorov Complexity とランダムネス

東工大・情報理工学　小林 孝次郎

1　はじめに

“ランダムネス”, つまり“ランダムである”という概念はよく使われる概念であるが，これをきちんと数学的に定義しようとすると，なかなか難しい．このための有用な道具の一つに Kolmogorov complexity がある．本解説では，この Kolmogorov complexity にもとづく“ランダムネス”の定義について述べる．Kolmogorov complexity は，アルゴリズム（計算手順，あるいはコンピュータ・プログラム）という概念を道具としていろいろなものが持つ本質的な情報の量を計ろうとするものであり，これに基づく“ランダムネス”の定義は，ある意味で一番きつい定義といえるであろう．

2　Kolmogorov complexity とは

我々は，“ランダムネス”を定義する対象としては，(有限）ビット列，つまり 0,1 からなる有限列のみを考えることにする．いま，次のような 3 つの長さ 1000 ビットのビット列を考えて見よう．

(1)　01 を 500 回ならべたビット列 x_1,

(2)　円周率πの小数点以下の値の 2 進表現の最初の 1000 ビット x_2,

(3)　物理乱数から生成した 1000 ビットのビット列 x_3.

直観的にいえば，この 3 つのビット列については，x_1,x_2,x_3の順にだんだん“ランダムネス”が増大している，といえるであろう．

直接この“ランダムネス”を定義しようとするかわりに，この 3 つのビット列の内容を電話で他人に伝えようとする時，どれぐらいの時間（いいかえれば電話料金）がかかるか，ということを考えて見よう．まず x_1については，単に“01 を 500 回並べたビット列”といえばよいから，極めて短い時間で伝えることができる．x_2についても，“円周率π（あるいは $4\ \tan^{-1} 1$）の小数点以下の値の 2 進表現の最初の 1000 ビット”と伝えればよい．それには x_1の場合に比べ少し余計に時間がかかるであろうが，それほど大変ではない．それに比べ x_3の場合は，その内容をそのまま電話口で読み上げるというのが，結局いちばんよい方法であろう．したがって x_1,x_2の場合に比べ，ずっと長い時間が必要であろう．つまり，ビット列を具体的に記述しようとするとき，x_1, x_2,x_3の順に記述が長くなることになる．

ビット列 x の Kolmogorov complexity は，この“ビット列の記述の長さ”というものに着目してビット列が含む本質的な情報の量とでもいうべき量をとらえようとするもので，

$$K(x) = x \text{の記述の長さの最小値}$$

として定義される．ただしこれがちゃんとした定義になるためには，“記述”という言葉をきちんと定義しなければならない．

次の 2 つの条件を満足するような Pascal プログラム Pを，**解釈プログラム**と呼ぶ．
（条件 1 ）Pはビット列を入力として受けとり，ビット列を出力として出して停止する．ただし入力によっては，停止しないでいつまでも計算を続けてもよい．
（条件 2 ）ビット列 dを入力として Pに与えた時に Pが停止すれば，dの後に 1 個以上のビットを付け加えたビット列を Pに与えると，Pは決して停止しない．

入力ビット列 dを Pに与えると Pがビット列 xを出力して停止する場合には，dは（解釈プログラム Pにおける）xの**記述**であるということにする．また dは（解釈プログラム Pにおける）**正しい記述**である，という．Pが停止しないときには，dは（解釈プログラム Pにおける）正しい記述ではない，という．条件 2 は，正しい記述の後に 1 個以上のビット列を付け加えると，もはや決して正しい記述にならない，ということを意味している．この条件を付け加えるのは，2 つの正しい記述 d,d'を並べて得られるビット列 dd'が与えられたとき，それから元の 2 つの記述 d,d'を復元することを可能にするためである．

Pが解釈プログラム，xがビット列であるとき，$K_{\text{Pascal},P}(x)$ を

$$K_{\text{Pascal},P}(x) = \text{解釈プログラム } P\text{における } x \text{の記述の長さの最小値}$$

と定義する．ただし解釈プログラム P における xの記述が存在しない場合には，$K_{\text{Pascal},P}(x) = \infty$ と定義する．

問題は，解釈プログラム Pとしてどのようなものを選ぶべきか，ということであるが，次の結果が成り立つことより，これは解決される．

定理 1 次の条件を満足するような解釈プログラム P_{U}が存在する：どのような解釈プログラム Pに対しても，すべての xに対して

$$K_{\text{Pascal},P_{\text{U}}}(x) \leq K_{\text{Pascal},P}(x) + c$$

がなりたつような定数 cが存在する．

この定理の条件を満足する解釈プログラム P_{U}を，**万能解釈プログラム**と呼ぶ．定理 1は，万能解釈プログラム P_{U} から作られる関数 $K_{\text{Pascal},P_{\text{U}}}(x)$ は，定数以下の差を無視することに

すれば，あらゆる解釈プログラム Pに対する関数 $K_{\mathrm{Pascal},P}(x)$ の中で最小である，ということを意味している．そこで，ある万能解釈プログラム P_{U}を選んで固定し，この P_{U}に対する関数 $K_{\mathrm{Pascal},P_{\mathrm{U}}}(x)$ を $K_{\mathrm{Pascal}}(x)$ で表すことにする．

$P_{\mathrm{U}}, P'_{\mathrm{U}}$が共に万能解釈プログラムであれば，ある定数 c が存在してすべての x に対して $|K_{\mathrm{Pascal},P_{\mathrm{U}}}(x) - K_{\mathrm{Pascal},P'_{\mathrm{U}}}(x)| \leq c$ が成り立つ．従って，定数の差を無視することにすれば，関数 $K_{\mathrm{Pascal}}(x)$ は，万能解釈プログラム P_{U}の選び方にはよらない．

Pascal以外のプログラミング言語に対しても同様に，$K_{\mathrm{C}}(x)$, $K_{\mathrm{lisp}}(x)$, $K_{\mathrm{assembler}}(x)$ などを定義することができるが，例えば Pascal と C の組合せに対し，次の結果がなりたつ．

定理 2 すべての x に対し

$$|K_{\mathrm{Pascal}}(x) - K_{\mathrm{C}}(x)| \leq c$$

がなりたつような定数 c が存在する．

プログラミング言語の他の組合せについても，同様の結果がなりたつ．従って $K_{\mathrm{Pascal}}(x)$ という関数は，定義には Pascal という特定のプログラミング言語を用いているものの，Pascal には依存しない普遍的な概念であることがわかる．そこでこの $K_{\mathrm{Pascal}}(x)$ を単に $K(x)$ で表し，ビット列 x の **Kolmogorov complexity** と呼ぶ．

3 有限列のランダムネス

ビット列 x が直観的な意味で“ランダム”なら，その記述は実質的に x の内容を直接指定することになり，どのように工夫しても，記述の長さをあまり小さくすることはできないであろう．一方もし x が“ランダム”でないなら，x はある意味での規則性なり冗長性なりを持っているであろうから，そのことを利用して x の短い記述が可能であろう．従って，x が“ランダム”なら $K(x)$ の値は大きく，そうでなければ $K(x)$ の値は小さいであろう．このような意味で，$K(x)$ は x の“ランダムネス”を表しているといえる．それでは次に，$K(x)$ はどの程度の大きさの量であるのかということを，もう少し具体的に調べてみよう．以後，ビット列 x の長さを n で表すことにする．

まず x の記述として，“x の内容をそのまま記す”というものがある．ただし入力 x をそのまま出力する Pascal プログラム P_1は，条件 2 を満足しないので解釈プログラムにはならない．そこで少し工夫が必要である．ビット列 x に対し，次の 3 つのビット列を順にならべて得られるビット列を $\mathrm{rep}(x)$ で表す：(1) n（x の長さ）の 2 進表現から 0 を 00 で，1 を 01 で置き換えて得られるビット列，(2) 1，(3) x．この $\mathrm{rep}(x)$ が入力として与えられれば x を出力して停止し，入力が $\mathrm{rep}(x)$ という形をしていなければいつまでも停止しない，という

Pascalプログラム P_2は，解釈プログラムである．この P_2を利用することにより，ある定数 cについて，$K(x) \leq n + 2\log_2 n + c$がすべての xに対してなりたつことが示せる．この結果は，$K(x) \leq n + \log_2 n + 2\log_2 \log_2 n + c$などの形に改良できる．

一方，長さ nのビット列のうち $K(x) \leq n - k$を満足するものの比率は，$1/2^{k-1}$以下である．このことは，P_Uの正しい記述のうち長さが $n-k$以下のものの個数が $1+2+2^2+\ldots+2^{n-k} < 2^{n-k+1}$を越えないことから直ちに得られる．例えば，$K(x) \leq n - 21$ であるような xの比率は $1/2^{20} = 1/1048576$以下である．従って，$K(x)$の値が nに比べて大幅に小さいような xは，“異常に短い記述を持つ”という意味で“非ランダム”である．例えば次の条件がなりたつ場合には，後の場合ほど xは“非ランダム”であると考えてよい：(1) $K(x) = n - \log_2 n$，(2) $K(x) = n - \sqrt{n}$，(3) $K(x) = n/2$，(4) $K(x) = \sqrt{n}$，(5) $K(x) = \log_2 n$，(6) $K(x) = \log_2 \log_2 n$．

xが何らかの規則性や冗長性を持つ場合には，$K(x)$の値は小さい．例えば xが 0101...01 という形のビット列である場合や，円周率πの小数点以下の値の2進表現の最初の nビットである場合を考えてみよう．このような場合には，xはその長さ nだけで完全に決まってしまう．従って，ある定数 cについて $K(x) \leq \log_2 n + 2\log_2 \log_2 n + c$がなりたち，$K(x)$の値は非常に小さい．

もう1つの例として，xに含まれる1の比率が 0.51 以上であるという場合を考えて見よう．長さが nのビット列は全部で 2^n個あるが，その中で上の条件を満足するものの比率は $e^{-(1.9999)(0.01)^2\sqrt{n}}$以下であることが示せる．従って，“そのようなものの中の何番目”という記述を使用することにより，$K(x) \leq n - 0.0002885\sqrt{n}$ であることがわかる．従ってこの場合も，$K(x)$の値はかなり小さい．

これに対し，長さ nのビット列 xをコイン投げなどによってランダムに生成すると，例えば $1 - 1/1048576$ 以上の確率で $K(x) \geq n - 20$ であるような xが得られる．

Kolmogorov complexity の応用として，$K(x)$の値を具体的に求めることにより乱数の検定を行なったり，$K(x)$の値が大きい xを具体的に求めることにより品質のよい乱数を生成する，というようなものが考えられる．しかしながら次の定理が示すように，$K(x)$の値を具体的に求めることは極めて困難であり，Kolmogorov complexity をそのような目的に利用することは，残念ながらできない．

定理 3 ビット列 xが与えられるとそれに対する $K(x)$の値を出力するようなアルゴリズム（例えばPascalのプログラム）は存在しない．

4 無限列のランダムネス

Kolmogorov complexity により，有限列のランダムネスを定義することができた．しかし，無限列のランダムネスが問題になることも多い．本節では，Martin-Löf ランダムネスと呼ばれる無限列のランダムネスの概念を紹介する．有限列の場合と同様，無限列としては無限ビット列のみを考える．

Xが無限ビット列の集合であるとき，0 または 1 を等確率，独立に繰り返してランダムに選ぶことにより生成した無限ビット列が Xに含まれる確率を，$\mu(X)$ で表す．例えば Xが 001 で始まるような無限ビット列の集合であれば，$\mu(X) = (1/2)^3 = 1/8$ である．また Xが，少なくとも 1 個の 1 を含み最初の 1 の後には必ず 00 が続く，というような無限ビット列の集合であれば，Xは 100,0100,00100, ... のいずれかで始まるの形の無限ビット列の集合であるから，$\mu(X) = 1/8 + 1/16 + 1/32 + \ldots = 1/4$ である．

無限ビット列の集合の無限列 $X_0, X_1, X_2, \ldots$ で，

- $X_0 \supseteq X_1 \supseteq X_2 \supseteq, \ldots$,
- $\mu(X_i) \leq 1/2^i$

という 2 つの条件を満足するようなものを，**しぼり込み**と呼ぶことにする．またしぼり込み $X_0, X_1, X_2, \ldots$ は，次の条件を満足するアルゴリズム Pが存在するとき，**アルゴリズムによって生成できる**という．

- Pに自然数 i を入力として与えると，Pはいつまでも動き続け，ことなる有限ビット列 $s_{i0}, s_{i1}, s_{i2}, \ldots$ を次々と出力してゆく．ただし，ある時点から後，全く何も出力しなくなってもよい．
- X_iは，Pに i を入力として与えたときの出力である有限ビット列 $s_{i0}, s_{i1}, s_{i2}, \ldots$ のいずれかで始まるような無限ビット列の集合になっている．

無限ビット列αは，アルゴリズムで生成できるようなどのようなしぼり込み $X_0, X_1, X_2, \ldots$ に対しても，ある i から後の $X_i, X_{i+1}, \ldots$ に含まれなくなるとき（つまり$\alpha \notin \bigcap_i X_i$であるとき），**Martin-Löf ランダム**であるという．そして Martin-Löf ランダムでないとき，**Martin-Löf 非ランダム**であるという．

いくつかの例で，直観的な意味でランダムでない無限ビット列が Martin-Löf 非ランダムであることを確かめてみよう．

まず$\alpha = a_1 a_2 \ldots$ が，010101... とか円周率πの小数点以下の値の 2 進表現のように，アルゴリズムによって計算できてしまう場合には，明らかにαは Martin-Löf 非ランダムであ

る．この場合 X_iとしては，$a_1a_2\ldots a_i$で始まる無限ビット列の集合を選べばよい．別の例として，αの中に0より1の方が頻繁に出現する場合を考えて見よう．例えば，αのどの長さの頭部においても1の比率が0.51以上であったとする．この場合は，長さが $f(i)$ 以下のすべての頭部において1の比率が0.51以上である，というような無限列の集合を X_iとして選ぶことにより，αが Martin-Löf非ランダムであることが示せる．ただし $f(i) = (i/0.0002885)^2$ である．

上の2つの例では，αのビットを具体的に求めることができる（1番目の例の場合），あるいはαの中の0,1の分布にくせがある（2番目の例の場合），ということを利用することによって，αをいつまでも含み続けるようなしぼり込み $X_0, X_1, \ldots$ をアルゴリズムによって生成することができた．しかしαが直観的な意味でランダムである場合には，このように利用できる特徴がなく，αをいつまでも含み続けるしぼり込み $X_0, X_1, \ldots$ を作るには，αのビットに関するある意味で無限の情報を知っていることが必要であり，そのようなしぼり込みをアルゴリズムによって生成することは，むずかしそうである．

Martin-Löf ランダムネスは，次の定理が示すように，Kolmogorov complexity によっても特徴づけることができる．

定理 4 無限ビット列αに関する次の2つの条件は同値である．

(1) αは Martin-Löf ランダムである．

(2) ある定数 c が存在して，任意の n に対して $K(\alpha_n) \geq n - c$ がなりたつ．ただしα_nはαの長さ n の頭部を表す．

5 ランダムな無限列の具体例

Xを Martin-Löf ランダムな無限ビット列の集合とすると，$\mu(X) = 1$ であることがわかっている．従って，“ほとんどすべての無限ビット列は Martin-Löf ランダムである”ということができる．しかし，Martin-Löf ランダムな無限ビット列が多いからといって，Martin-Löf ランダムな無限ビット列の具体例を示すのが容易であるとは限らない．

G. Chaitin は，$\Omega = \Sigma \{2^{-|d|} \mid d$ は P_Uの正しい記述 $\}$ という式で定義される実数Ωの2進表示である無限ビット列が Martin-Löf ランダムであることを示した．Ωは，$\mu(\{\alpha \mid \alpha$ は入力として P_Uに与えると P_Uが停止するような有限ビット列で始まる $\})$ という値でもあるから，“ランダムな入力を P_Uに与えたときに P_Uが停止する確率”といいかえてもよい．アルゴリズムによって計算できる無限ビット列は Martin-Löf ランダムではないから，$\Omega = \Sigma \{2^{-|d|} \mid d$

は P_Uの正しい記述 $\}$ というΩの定義は，値を定義してはいるが，その2進表示を計算するアルゴリズムまでは示していない定義，であるということになる．

Chaitin はこのΩをさらに変形して，exponential diophantine equation と呼ばれる整数論の問題に Martin-Löf ランダムネスが現れることを示した．

自然数を値にとる変数 $x_1, x_2, \ldots$ と自然数を表す定数から，加算+，乗算・，冪乗演算を有限回適用して得られる2つの式 $f(x_1, x_2, \ldots, x_n)$，$g(x_1, x_2, \ldots, x_n)$ を等号=でつないで得られる

$$f(x_1, x_2, \ldots, x_n) = g(x_1, x_2, \ldots, x_n)$$

という形の方程式を，exponential diophantine equation という．例えば

$$32^{(x_1+x_2^{x_3}x_4)^2} + x_1^{x_2+1} = x_3^{16} + (x_1 + x_2)^{x_4^2+1}$$

はその例である．フェルマーの "定理" は，exponential diophantine equation

$$(x_1 + 1)^{x_4+3} + (x_2 + 1)^{x_4+3} = (x_3 + 1)^{x_4+3}$$

が解を持たない，という主張にほかならない．

exponential diophantine equation が解を持つか否かを決定する問題は大変むずかしく，それを一般的に解くアルゴリズムは存在しないことが知られている．exponential diophantine equation が有限個の解を持つか無限個の解を持つかを決定することは，さらに難しい問題である．Chaitin は，1つのパラメタ n と 16962 個の変数 $x_1, \ldots, x_{16962}$ に関する exponential diophantine equation

$$f(n, x_1, x_2, \ldots, x_{16962}) = g(n, x_1, x_2, \ldots, x_{16962})$$

で，各 n に対し，この方程式が無限個の解を持てば $a_n = 1$，有限個の解を持てば $a_n = 0$，という条件で定義した無限ビット列 $a_1 a_2 \ldots$ が Martin-Löf ランダムになるようなものを，具体的に示した．この方程式は，左辺が約 470,000 文字，右辺が約 420,000 文字，合計約 890,000 文字の，巨大なものである．

参考文献

1. M. Li, P. Vitányi, Kolmogorov complexity and its applications, in "Handbook of Theoretical Computer Science", vol. A (ed. by J. van Leeuwen), 187-254, Elsevier and The MIT Press, 1990 （邦訳：広瀬 健 他監訳，コンピュータ基礎理論ハンドブック I，丸善（株），185-262，1994）．

2. M. Li, P. Vitányi, An Introduction to Kolmogorov Complexity and its Applications, Springer-Verlag, 1993.