統計数理(2006)
第 54 巻 第 2 号 511–523

特集「予測と発見」
[研究詳解]

計算論から見たランダムネス

渡辺 治†

(受付 2006 年 1 月 4 日; 改訂 2006 年 5 月 8 日)

要　　旨

計算論から見たランダムネスの研究として，無限列のランダムネスを定義したマーチンレフ・ランダムネス，そして有限列のランダムネスを定義したコルモゴロフ・ランダムネスと，その関係について解説する．また，コルモゴロフ記述量の研究の応用として，最近提案された情報距離とその簡易版を紹介する．

キーワード：計算論，コルモゴロフ記述量，無限列のランダムネス，有限列のランダムネス，情報距離．

1. はじめに

莫大な情報を扱うにはコンピュータによる「計算」が不可欠であることは多くの人が認めるところである．けれども，コンピュータは道具であり，あくまで統計的な分析手法を実現する手段である，と考えている人も少なくないだろう．本稿では，情報の分析において「計算」が本質的な役割を果たす場合があることを示す．具体的には，計算という観点で見ることにより，ランダムネスの一面が明らかになり，ランダム性・規則性・類似性の概念を統一的に議論することができることを紹介する．なお，本稿でいう「計算」とは，単なる数式による計算だけではなく，抽象的にはアルゴリズム，具体的にはプログラムとして記述される広い意味での計算のことである．

ランダムという概念は確率や統計の基本的な概念である．むしろあまりに基本的すぎて，ランダムネスは公理のように扱われている感がある．それに対し，Turing や Church 等により「計算」という概念が明確にされ，コンピュータ・サイエンスが始まった頃，Solomonoff，Kolmogorov，Chaitin，Martin-Löf 等の研究者が，計算という立場からランダムネスを明確にすることに成功した．本稿では，その研究の一端を筆者の観点から解説する．最後の節では，こうした研究の実際への応用例として，最近注目を集めている汎用で簡便なデータ間の類似度の測り方を紹介する．マーチンレフ・ランダムネスやコルモゴロフ記述量についてよくご存知の方，あるいは，取りあえずは新しい技術を知りたい方には，途中を飛ばして最後の節から読まれることをお勧めする．

本稿では 0 と 1 からなる列—二進列—を対象とする．以下では単に「列」といった場合には二進列をさすものとする．与えられた列がランダムであるか否かを決める基準はあるだろうか？「そのような基準を問うこと自体意味がない」というのも 1 つの答えである．たとえば，「7 ビットの長さの列をランダムに生成する」というのはよい．けれども，「0000000 と 001101 のどちらがランダムか？」という問いは意味がない．というのも，ランダムに生成した場合，

† 東京工業大学 情報理工学研究科：〒152–8552 東京都目黒区大岡山 2–12–1

どちらも同じ確からしさで出現するからである，という考え方である．けれども，前者の列には規則性があり，後者の列にはそれほど規則性がない(ランダムに近い)ようにも思える．この直感を数学的に表わす方法はないだろうか？

規則性のある列は，その規則性を使って圧縮できるはずである．たとえば 3000 個の 0 の列は，よく 0^{3000} と略記されるが，これはまさに規則的だから，このような略記が可能なのである．3000 ビットの列に規則性がなければ，それを表わすには，その 3000 ビットをそのまま書かなければならないだろう．与えられた列の規則性・ランダム性を測るには，それを出来る限り圧縮し，その圧縮の程度を用いるのがよいように思われる．この方法を数学的に定義したのがコルモゴロフ記述量である．圧縮には，圧縮する計算と復元する計算の 2 種類の「計算」が関与してくる．コルモゴロフ記述量は，復元という計算を考えることによって，ランダムネスを議論する方法である．このコルモゴロフ記述量を用いて有限列のランダム性を定義したのが(有限列に対する)コルモゴロフ・ランダムネスと呼ばれる概念である．

このような直感的な議論だけでは研究として先に進まない．圧縮性をもとに規則性・ランダム性を測る方法を定義したとき，その意味や妥当性を厳密に議論していかなければならない．そのために本稿では，まず無限列のランダムネスに対する数学的な議論を考える．無限長の二進列であれば，一般の確率論でもランダムネスの特質や検定法が深く研究されている．無限列に対してはランダムネスの数学的な共感が得られている，といってもよいだろう．Martin-Löf は，計算論という道具を用いて，その共感を数学的に厳密化することに成功した．それが(無限列に対する)マーチンレフ・ランダムネスである．一方，ランダム無限列に対し，その最初の n ビットの列(先頭部分列)は，n が十分大きいときにはランダム列になっていると見てもよいだろう．まさに，その関係がマーチンレフ・ランダムネスとコルモゴロフ・ランダムネスの間に成り立っているのである．

無限列 ω がマーチンレフ・ランダム
$\iff$ そのすべての先頭部分列がコルモゴロフ・ランダム

本稿では，これがコルモゴロフ・ランダムネスやコルモゴロフ記述量の妥当性の根拠である，という観点から関連の研究結果を紹介していく．

ランダムネスの定義に関する議論に対しては，哲学的な議論で興味深いが役に立たないもの，という印象を持たれる場合がある．しかし，ランダムネスの意味を厳格に議論することは，その正反対にある規則性や，規則性を司る情報の本質を議論していることでもある．そこで得られる知見や方法論は実際にも役立つはずである．その一例として，本稿の最後に，コルモゴロフ記述量の研究から発生してきた，データ間の類似性を測る簡便で汎用な方法を紹介する．

この手法は，圧縮という通常よく用いられている「計算」を使うことで，データ間の類似性を測り，それにより，多数のデータの分類を発見する方法である．その大きな特徴は，ファイルとして表現されるあらゆるデータに対し使用できる万能(汎用)な方法であるという点である．「計算」は捉えにくい反面，うまく利用することができれば「汎用性」を期待することができるのである．ここで紹介する方法も，その意味や可能性についてはまだまだ未解決の点が多いが，将来有望な方法の 1 つといえるだろう．

本稿を通して，技術的な内容から，定理などの歴史的背景まで，この分野の代表的な教科書である Li and Vitányi (1993)を参考にした．ただし，2 節の内容の原典は Martin-Löf (1966)である．また，4 節の説明では，Cilibrasi and Vitányi (2005)の枠組みに従った．

2. 無限列のランダム性：マーチンレフ・ランダムネス

マーチンレフが考え出した無限列のランダム性を議論する方法について解説する．本稿では，

無限の二進列を ω と記述する．また，その先頭の n ビット目までを $\omega_{1:n}$ と書くことにする．

与えられた無限の二進列 ω に対して，そのランダム性を統計的にテストすることを考えてみよう．無限列すべてを見ることはできないので，我々はその一部の有限列，典型的には先頭部分列 $\omega_{1:n}$ を n を増加させながら調べることになる．たとえば，1 の出現する回数(これを $\#_1(\omega_{1:n})$ で表わす)に注目してみる．この回数は平均で $n/2$ だが，それが平均より大幅に大きくなるようであればランダム性が薄いと考えてよい．具体的には，テスト A として，$\#_1(\omega_{1:n}) - n/2$ と $\sqrt{n}$ の比の増加性を見るテストを考える．この比がどんどん大きくなっていくようだと「ランダム列でない」と判断するのである．このテストは悪くはない．もちろん，テスト A だけでランダムネスを判定することはできない．けれども，テスト A で「ランダム列ではない」と棄却される無限列は非常に稀で，ランダムに生成された列であれば確率 1 で，このテストに合格するからである．つまりランダムネスのテストとしては「安全」である．一方，条件をもう少し厳しくし，$\#_1(\omega_{1:n}) - n/2$ と $n^{1/4}$ の比の増加性を見るテストは，ランダムネスのテストとしては不適切である．そのテストに合格する無限列の確率は 0，つまり，特殊な列しか合格しないからである．

このように，先頭部分列 $\omega_{1:n}$ に対して漸近的にテストすることができ，ランダムに生成された列であれば確率 1 で合格するようなテストを，我々はランダムネスに対する「安全な」統計テストと考える．Martin-Löf は，この考え方を次のように形式化したのである．

定義 1. 無限二進列から自然数への関数 δ で，次の条件を満たすものを(ランダムネスの)列テスト (sequential test) (の評価関数)という．

1. 有限二進列上の関数 γ を用いて $\delta(\omega) = \limsup_{n\to\infty} \gamma(\omega_{1:n})$ と定義できる．
2. 任意の $m > 0$ に対して，$\Pr_\omega[\delta(\omega) \geq m] \leq 2^{-m}$ が成り立つ．

実際の列テストは，この評価関数 δ に対して $\delta(\omega) = \infty$ か否かを判定することである．$\delta(\omega) < \infty$ のときは，列 ω が列テスト δ に合格したという．反対に $\delta(\omega) = \infty$ のときは列テストにより棄却されたという．

たとえば，先のテスト A のためには，

$$\gamma_A(\omega_{1:n}) \;=\; \log \frac{\#_1(\omega_{1:n}) - n/2}{\sqrt{n}}$$

と定義する(なお，本稿を通じて $\log n$ 底は 2 とする)．すると γ_A から定義される δ_A によるテスト，すなわち $\delta_A(\omega) = \infty$ か否かのテストが，テスト A にちょうど一致している．一方，任意の $m > 0$ に対して，適当な n を考えれば $\Pr_{\omega_{1:n}}[\gamma(\omega_{1:n}) \geq m] \leq 2^{-m}$ を示すことができる．また，一般に任意の $\rho > 0$ に対して

$$\exists n \left[\Pr_{\omega_{1:n}} [\gamma(\omega_{1:n}) \geq m] \leq \rho \right] \implies \Pr_{\omega}[\delta(\omega) \geq m] \leq \rho$$

が成り立つ．これらから δ_A が列テストの条件を満たすことが直ちに示せる．

補足．定義の条件 2 の確率の上界 2^{-m} は厳しい制限のように見える．けれども，元々のテストでの棄却確率が 0 であれば(n の増加とともに $\omega_{1:n}$ が棄却される確率が 0 に収束するのであれば)，γ を少し増加しにくい関数にすることで，多くの場合，この条件を成り立たせることができる．たとえば，テスト A の例でも，比を直接使うのではなく，その対数をとったものを γ_A の定義に使用している．これにより，比較的簡単に $\Pr_{\omega_{1:n}}[\gamma(\omega_{1:n}) \geq m] \leq 2^{-m}$ を示すことができるのである．

ランダムネスを調べる列テストをどの程度用意すればよいのだろうか？ 1 つや 2 つでは心

もとない．そこで自然に思いつくのは「すべての」列テストを用いる方法である．すべての列テストに合格する列をランダムと考えよう，という発想である．自然な考え方ではあるが，残念ながらランダムネスの定義には使えない．すべての列テストに合格する列は存在しないからである．無限二進列の全体は実数の濃度を持つが，列テストも同じく実数濃度だけある．しかも，各無限二進列に対し，それを棄却する列テストが定義できてしまうのである．

ここで「計算」という概念が重要になってくる．マーチンレフは「すべての計算可能な」列テストを考えた．計算可能な列テストとは，上記の γ を計算するプログラムが存在するような列テストのことである．マーチンレフは次の定義を提唱したのである．

$$\omega \text{ がランダム} \iff \omega \text{ はすべての計算可能な列テストに合格する} \tag{2.1}$$

以下ではこれをマーチンレフ・ランダムネスと呼ぶことにする．

マーチンレフ・ランダムな列のよい点は，ランダム列に対して我々が期待するすべての統計的性質を by definition として持つ点である．

その代わり，ここで重要なのは，そのような性質を持つ無限列が十分存在することである．通常の意味でランダムに無限列を生成したとき，プログラムの個数は可算無限個しかない，そこで「すべての計算可能な列テスト」を考えても棄却されない列は存在する．実際には，無限二進列は，確率 1 でマーチンレフ・ランダムにならなければ意味がない．幸い，その性質はマーチンレフにより証明されたのである．

定理 1.

$$\Pr_{\omega}[\ \omega \text{ はマーチンレフ・ランダム}\ ] = 1.$$

証明の考え方を簡単に説明する．すべての計算可能な列テストを考えたのであるが，実は，それを 1 つのテストに集約することができる．それを万能列テストという．この万能列テストは，任意の計算可能な列テストを，部分テストとして含むようなテストなのである．

無限個の列テストに合格する列を考えるのは難しいが，万能列テストとはいえ，列テストが 1 つに絞られれば考えやすい．実際，その万能列テストも列テストの 1 つであるから，列テストの定義の条件 2 より，棄却される無限列の確率が 0 であること，つまり合格する無限列の確率は 1 であることは明らかである．

最後に万能列テストの構成法について，少々大ざっぱだが考え方を述べよう．万能列テストの本体は，プログラミング言語のインタープリタと同じ構造である．列テスト(正確には γ)のプログラムを読みながら，その実行を模倣するプログラムなのである．また，すべての列プログラムに番号を付け，それを小さい順に生成する仕組みも必要である．万能列テストが各 n に対して先頭部分列 $\omega_{1:n}$ から γ を計算するときには，1 番目から n 番目までの列テストのプログラムを，その列テストインタープリタに入れて実行し，その最大値を γ の値として使う．これにより，どれかのテストで棄却される場合には，この万能テストでも棄却されるのである．

普段の研究で，何らかのプログラミング言語のインタープリタを使っている方も多いと思う．それは道具としての使用である．けれどもインタープリタという概念は，単なる道具ではなく，計算論では非常に重要な概念なのである．それがランダムネスの定義でも鍵となっていたのである．

3. コルモゴロフ記述量

マーチンレフ・ランダムネスは，無限列に対してランダムネスを定義する方法である．それに対し，有限列に対するランダムネスを数学的に定義するには，計算論的なもう一つの概念―コ

ルモゴロフ記述量―が必要である．この節では，以降の節のためにコルモゴロフ記述量(正確には自己分離型コルモゴロフ記述量)の定義と基本性質について解説する．

3.1 コルモゴロフ記述量の定義

コルモゴロフ記述量を定義するには，まず計算モデルを定めなければならない.「計算モデル」というと大層な物のように響くが，ここでは計算を表わす方法―アルゴリズムの表現法―と考えればよい．計算論では，チューリングが導入した「チューリング機械」を用いるのが標準的だが，とくにそれにとらわれることはない．たとえば，妥当なプログラミング言語や，基本素子だけからなる計算回路図(の表わし方)も，計算の表現法として用いることができる．ここでは，多くの読者に馴染みのある C 言語を我々の表現法と定めることにしよう．

定められた表現法により計算を記述したものをプログラムという．我々の表現法では，プログラムとは，まさに C 言語で書かれたプログラムのことである．与えられた文字列のコルモゴロフ記述量とは，その列を生成する最短のプログラム長である．ただし，正確な定義には，もう少し厳密に議論する必要がある．

まず多少の約束をしておく．最初は我々の使用する C 言語についての約束である．C 言語には文字列定数があるが，二進列を対象とする特別の二進列定数というものを導入する．たとえば，`'011000101110101'` のように 15 ビットの二進列を表わす．ここに書ける二進列は少し限定されるのだが(後述)，それ以外は文字列定数 `"011000101110101"` と同じように使える．ただし，プログラム全体を二進列に変換するときに効率的に変換される点が異なるのである(後述)．

プログラムの書き方にも少し制約を導入する．コルモゴロフ記述量に用いる(C 言語の)プログラムには，入力無しか，あるいは 1 つの二進列を(引数として)入力するものしか考えないことにする．また，出力も 0 か 1 (の列)しか許さないことにする．引数を 1 つ取るプログラム P に対して列 x を与えたとき，プログラム P が停止するまでに出力した記号列を $\mathtt{P}(x)$ と表わす．引数を取らない(入力無しの)プログラムの出力は $\mathtt{P}()$ と書く．もし停止しない場合には，$\mathtt{P}(x)$ や $\mathtt{P}()$ は未定義であるとする．

次に文字列とその長さについての基本的な概念を準備する．コンピュータ・サイエンスでは一般に，文字列 x の長さ(文字数)を $|x|$ と記述するが，使用できる文字種に多少の曖昧さが残る．ここでは二進列のみを考えているので，この曖昧さはなく，$|x|$ は明確に二進列のビット数となる．もしも x が英字や記号からなる列の場合には，二進列に符号化して考える．C 言語で書かれたプログラム自身も，各記号を 7 ビットで表わす符号(いわゆる ASCII 符号) で二進列に符号化されているものとする．

この単純な方法では，元の文字列長のつねに 7 倍のビット長になってしまう．しかし，もっと効率よく符号化したい場合もある．そこで二進列定数だけは特別に効率よく符号化することにする．具体的には，たとえば

$$\text{'01100001001010001'} \xrightarrow{\text{符号化}} \underbrace{01100000}_{\text{記号 ' のコード}}\ 01100001001010001\ \underbrace{01100000}_{\text{記号 ' のコード}}$$

といった具合に，二進列の部分だけは「そのまま」埋め込むのである．けれども，そうすると 7 ビットに符号化された前後の記号 ' との分離ができなくなる場合がある．そこで，左から見ていったときに，何らかの方法で終わりが判別できる二進列のみを二進列定数として書けるものとする．符号の言葉でいえば語頭符号 (prefix code) になっているものだけを許すことにする．プログラムの長さとは，このように二進列に符号化された上での，プログラムのビット長とする．たとえば，プログラム P に対して，このビット長を $|\mathtt{P}|$ と表わすことにする．

補足．語頭符号について．一般の二進列 x を語頭符号にするにはいろなやり方があるが，こ

こでは単純に，先頭に x のビット長を埋め込む方法を採用しよう．たとえば，010010011011 は 12 ビットなので，

$$010010011011 \overset{\text{語頭符号化}}{\longrightarrow} \underbrace{\underline{01100001}}_{1}\ \underbrace{\underline{01100010}}_{2}\ \underbrace{\underline{01101010}}_{\text{記号：のコード}}\ 010010011011$$

のように，ビット長と区切り記号 : を 7 ビット符号化したものを本体 x の前に付けるのである．こうすれば，左から二進列を見ていって 12 という情報を取り出せるので，x を「そのまま」書いたとしても，それがどこで終わるかがわかるだろう．以下では，このような符号化したものを bin(010010011011) と書くことにする．二進列定数には，このように語頭符号化したもの，それにそれを適宜拡張したものを許すことにする．

以上の約束のもと，列 x に対し $\mathrm{K}(x)$ を次のように定義する．

$$\mathrm{K}(x) \ = \ \min\{|\mathtt{P}| \ : \ \mathtt{P}() = x\}.$$

この量がコルモゴロフ記述量である．この定義は，プログラムが単独で x を生成する場合のものだが，入力 y に対して(それを手がかりに)x を生成する場合への拡張も考えておく．具体的には次のように定義する．これは y という情報が与えられたという条件のもとでの記述量なので条件付きコルモゴロフ記述量，あるいは相対的コルモゴロフ記述量と呼ばれている．

$$\mathrm{K}(x|y) \ = \ \min\{|\mathtt{P}| \ : \ \mathtt{P}(y) = x\}.$$

補足．ここで定義したコルモゴロフ記述量は，正確には自己分離型コルモゴロフ記述量(self-delimiting Kolmogorov complexity)と呼ばれている．プログラム(を二進列に符号化したもの)を左から見ていったとき，終わりが一意に定まるからである．単純なコルモゴロフ記述量に比べ，定義に多少気を使う必要があるが，きれいな議論を展開することができる．

3.2 コルモゴロフ記述量の基本性質

まず具体的な列に対してコルモゴロフ記述量がどの程度になるのかを見てみよう．たとえば「はじめに」で例に使った 3000 個の 0 の列 0^{3000} の記述量を評価してみる．この列は，次のようなプログラム P により生成される．

```
P = int main(){int i;for(i=0;i<3000;i++)printf("0");printf("\n");}
```

このプログラムは空白も入れて 63 文字，すなわち長さは $63 \times 7 = 441$ ビットである．したがって，コルモゴロフ記述量は $\mathrm{K}(0^{3000}) \leq 441 << 3000$ であり，列の長さ 3000 に比べてかなり小さい．これは 0^{3000} が規則的だったからである．

実際に我々が関心を寄せるのは逆の状況，すなわち，ランダムな列 x の特徴付けである．記述量 $\mathrm{K}(x)$ が大きな列 x を「非規則的」と考えるのはよい．けれども，どの程度の大きさを考えればよいだろうか？そのためには $\mathrm{K}(x)$ の上界を知っておく必要がある．

多くの計算モデルには，列 x を生成する「明らかな」方法がある．C 言語でも，たとえば printf（"010011"）とすれば 010011 が出力される．ただし，厳密に言えば，通常の文字列定数は冗長に符号化されるので二進列定数を使う．具体的には，n ビットの列 x は次のようなプログラム P で生成できる．

$$\mathtt{P} \ = \ \texttt{int main()\{printf('bin}(x)\texttt{');\}}$$

このプログラムのビット長 $|\mathtt{P}|$ は $|\mathtt{P}| = n + 7(\log n + 25 + 1) = n + 7\log n + 182$ である（正確には $\lceil \log(n+1) \rceil$ だが単純に $\log n$ と書くことにする）．以上の考察から，次のような記述量の上界が成り立つことは明らかだろう．

定理 2. 任意の列 x に対して次の不等式が成り立つ．

$$\mathrm{K}(x) \ \leq \ |x| + c_1 \log |x| + c_0.$$

ただし，c_0，c_1 は x に依存しない定数である．

補足．上の説明では定数 c_1 は 7 だったが，これは語頭符号化に工夫を加えれば限りなく 1 に近づけることができる．けれども 1 以下にはできないことも知られている．

コルモゴロフ記述量に関する重要な性質として，その計算不可能性を示す．

定理 3. 与えられた x に対して $\mathrm{K}(x)$ を計算するプログラムは存在しない．

証明． コルモゴロフ記述量を計算するプログラムが存在したとして矛盾を導く．そのプログラムを $\mathtt{A}$ とする．

パラメータとして与えられる数 n と m に対して，長さ n の列の中で最初に $\mathrm{K}(x) > m$ となる x を出力するプログラムを考える．「最初に」を決めるには順序を決める必要があるが，ここでは辞書式順序を用いよう．たとえば，長さ 3 の列では $\mathtt{000}$，$\mathtt{001}$，$\mathtt{010}$，$\mathtt{011}$，$\ldots$ の順序で検査するのである．プログラムを $\mathtt{A}$ を使えば $\mathrm{K}(x)$ が求まるので，その値が最初に m を超えるものを出力すればよい．このプログラムを $\mathtt{B}_{n,m}$ とする．（数 n，m はプログラム中では定数である．）

プログラム $\mathtt{B}_{n,m}$ の長さについては，適当な定数 $c > 0$ に対して

$$|\mathtt{B}_{n,m}| \ \leq \ 7(\log n + \log m) + c$$

が成り立つ．n，m は最初に適当な変数に代入して使えば，プログラム中で 1 度書けばすむので，それぞれ $7\log n$，$7\log m$ ビットの記述量になる．あとはプログラム $\mathtt{A}$ の記述も含めて固定なので定数で抑えられるからである．

さて，プログラム $\mathtt{B}_{n,m}$ が出力する列 $x_{n,m}$ を考えよう．$x_{n,m}$ はプログラム $\mathtt{B}_{n,m}$ により出力されるので，コルモゴロフ記述量の定義から $\mathrm{K}(x_{n,m}) \leq |\mathtt{B}_{n,m}|$ が成り立つ．一方，プログラム $\mathtt{B}_{n,m}$ の計算を考えると，$\mathrm{K}(x_{n,m}) > m$ でなければならない．したがって，

$$m \ < \ \mathrm{K}(x_{n,m}) \ \leq \ |\mathtt{B}_{n,m}| \ \leq \ 7(\log n + \log m) + c$$

が成り立つ．しかし，たとえば $m = n$ で十分大きな m に対しては，両端の関係は成り立たない．□

このように，残念ながらコルモゴロフ記述量は一般には計算できないのである．実は，単に計算不可能なだけでなく近似もできない．様々な近似の基準が考えられるが，妥当な基準に対しては近似不可能性の証明ができてしまうのである．

4. 有限列のランダム性：コルモゴロフ・ランダムネス

コルモゴロフ記述量を使って，有限列に対するランダムネスの定義を与えよう．パラメータ $k \geq 0$ に対して，

$$x \text{ が } k\text{-ランダム} \iff \mathrm{K}(x) \geq |x| - k \tag{4.1}$$

と定義する．適当な k に対する，この k-ランダム性でランダムネスを定義するのである．この k-ランダム性を総称してコルモゴロフ・ランダムネスと呼ぶことにする．

補足．議論をやりやすくするためにパラメータ k を導入した．けれども「そんなのは潔くない」というのであれば $k=0$ と固定して考えてもよい．記述量が大きい方が規則性が低いという観点からすれば，k は小さい方が強いランダムネスの定義になる．

コルモゴロフ・ランダムネスの妥当性について，いくつかの観点から述べてみよう．

十分大きい＆十分量がある

任意の長さ n を固定し，長さ n の列について考えることにする．コルモゴロフ記述量の上界は $n+c_1\log n+c_0$ だった．$c_1\log n+c_0$ の項は，十分大きな n を考えると無視できるほど小さいので，上界は，ほぼ n と考えてよい．我々の k-ランダムを決める基準 $n-k$ は，その上界に近い大きさである．

ランダムな列は多くなければならない．ランダム列は，直感的には特定の偏りのない「普通の列」なのである．ある特定の限られた列であれば，そこには何らかの特徴(つまり偏り)が存在する．したがって,「普通の列」の集合であるランダム列の集合は大きくなければならない．では k-ランダムな列は多いのだろうか？簡単な議論から次のような下界が導ける．

定理 4. 適当な定数 $c_2>0$ に対して,

$$k\text{-ランダムな列の数} \;\geq\; 2^n-2^{n-k-c_2} \;=\; 2^n\left(1-\frac{1}{2^{k+c_2}}\right).$$

証明． 任意の t に対し，$\mathrm{K}(x)<t$ となる(長さ n の列の)x の集合を J_t とする．J_t の要素各々に対しては，それを生成する長さ t 未満(つまり $t-1$ 以下)のプログラムが存在する．もちろん，各プログラムは異なる．では，長さ $t-1$ ビット以下のプログラムは何個存在するだろうか？

長さ $t-1$ 以下の二進列は高々 2^t 個しか存在しない．それらが全部プログラムの符号になっていたとしても，プログラム数は高々 2^t である．さらに，C プログラムでは，必ず `main` (これで 28 ビット)というキーワードを先頭に持つ，などの制約があるため，二進列全部がプログラムには対応しない．そのため，長さ $t-1$ 以下のプログラム数は，適当な定数 $c_2\geq 10$ に対して 2^{t-c_2} 以下とみてよい．

したがって，長さ t 未満のプログラムのすべてが各々異なる J_t の要素を生成したと仮定しても,(J_t の個数)$\leq 2^{t-c_2}$ である．ここで $t=n-k$ とする．J_{n-k} の要素でないものが k-ランダムな列である．一方，J_{n-k} の要素数は高々 2^{n-k-c_2} 個．以上から定理の関係式が導かれる．□

いま $k=0$ の場合を考え，仮に $c_2=10$ だったとしよう．すると，長さ n の列の総数 2^n に対して，その $(1-1/2^{10})=0.99902\cdots$ が 0-ランダムなのである．

計算モデルの妥当性

我々は気楽に C 言語を計算の表現に選んでしまったが，その選択でよかったのだろうか？他のもっと効率的な計算表現ならば，プログラムも短く記述でき，記述量も少なくてすむかもしれない．そのため C 言語とは「k-ランダム」の意味が大きく異なるかもしれない．

実は，そのような差は無視できる程度である．仮に記述効率の良さそうな計算表現法 W があったとしよう．ただし，その表現法で書いた W 流プログラム(を二進列に符号化したもの)は(我々が仮定している)語頭符号になっているものとする．その場合，任意の列 x に対して,

それを出力する最短プログラムを，それぞれ C 言語と表現法 W で考えると，

$$\text{最短 C プログラム長} \leq \text{最短 W 流プログラム長} + c_{\mathrm{W}}$$

が成り立つ．ただし，c_{W} は x に依存しない適当な定数である．

この関係から，表現法 W で考えても，C 言語による我々の記述量に比べコルモゴロフ記述量は定数程度しか小さくならないことがわかる．つまり，定数の差を除いては，我々の C 言語による記述量は最適なのである．

上記の関係の理屈を述べよう．W 流プログラムを解釈実行する C プログラムを作ることができる．いわゆる W 流プログラムのインタープリタである．そのインタープリタを使えば，任意の W 流プログラムから，それと同じことをする C プログラムが作れる．長さもインタープリタの大きさ程度（これが定数 c_{W}）しか増えない．したがって上の関係が成り立つのである．

無限列のランダムネスとの整合性

無限列に関しては 2 節で紹介したランダムネスの定義――マーチンレフ・ランダムネス――があり，今のところ妥当なものとして受け入れられている．このマーチンレフ・ランダムネスに対し，次の定理に示すように，我々の k-ランダム性は密接に関連しているのである．

定理 5.

$$\exists k \geq 0,\ \forall n \geq 0\ [\omega_{1:n} \text{ が } k\text{-ランダム}] \iff \omega \text{ がマーチンレフ・ランダム}.$$

補足．この美しい定理の発見者・証明者に対しては，いろいろな見解がある．詳しくは Li and Vitányi (1990，1993) を参照して頂きたい．

すべての先頭部分列がランダムであれば，その極限である無限列がランダムであると考えるのは自然である．また，その逆も納得できる．それを実際に示した上の定理は，マーチンレフ・ランダムネスとコルモゴロフ・ランダムネスの強い結びつきを示している．したがって，もしマーチンレフ・ランダムネスが無限列に対する正当なランダムネスの概念と認められるのであれば，この定理は，コルモゴロフ・ランダムネスが有限列のランダムネスを定義するのに「十分妥当な」概念であることの証拠となるだろう．

補足．「無限列がランダムである $\Longrightarrow$ すべての先頭部分列がランダム」には違和感を感じられる方がいるかもしれない．無限列がランダムであっても，「すべて」の先頭部分列がランダムである必要はないからである．最初の数ビットは例外的に非常に規則的な場合があるかもしれない．しかし，そのような場合でも，パラメータ k を十分大きく取ることで，例外的な部分も含め「すべて」の先頭部分列を k-ランダムとみなせるのである．

5. コルモゴロフ記述量の応用？

有限列のランダムネスを k-ランダム性のようにきれいな形で表わせる（正確には，表わすのが妥当である）ということは，我々のコルモゴロフ記述量が，同様の規則性・不規則性を表わす尺度の中でも正統派であることを示している．では，このコルモゴロフ記述量を何かに利用できないだろうか？ここではコルモゴロフ記述量の応用を 1 つ紹介する．これはデータ間の類似度を測る方法の提案である．計算不可能なコルモゴロフ記述量の応用，というと奇異に感じられるかもしれない．正確には，コルモゴロフ記述量から導き出された「考え方」を応用する話である．

最近，Vitányi を中心とするグループが，データの差異を測る非常に汎用な尺度を提案し，そ

れを用いて実際に様々なデータを分類して注目を集めている．その手法にそって分類するのを助けるソフトも公開されている Cilibrasi (2003)．この尺度はコルモゴロフ記述量の研究に裏付けられた尺度である．この尺度について，Cilibrasi and Vitányi (2005)で述べられている枠組みに従って説明する．

いくつかのデータがコンピュータ上のファイルとして与えられているとする．それらのデータファイル間の差を表わす「距離」を定義したい．ここでは各データファイルを各々 1 つの二進列とみなすことにする．一般にデータファイルは 1 つの文字列とみなすことができる．たとえば，数値実験の結果のファイルであれば，0 から 9 までの数字＋小数点＋区切りのカンマ＋改行記号，からなる文字列である．また，英文ファイルであれば，英字アルファベット(大文字，小文字)＋カンマ等，からなる文字列である．さらに，それらは二進列に変換されていると考え，各データファイルは(非常に長い)1 本の二進列とみなすのである．したがって，以下では，データファイルと二進列を同一視し，二進列間の類似度を測る尺度を考えることにしよう．

Bennett et al. (1998)は，与えられた 2 つの二進列 x と y の差を測る尺度 $\mathrm{E}(x,y)$ を導入した．これは条件付きコルモゴロフ記述量を用いると

$$\mathrm{E}(x,y) \;\approx\; \max\{\mathrm{K}(x|y), \mathrm{K}(y|x)\}$$

と近似できる(本稿では $\approx$ で，両辺が，その対数程度の差以内でほぼ一致していることを意味する)．そこで多少大ざっぱだが，右辺を $\mathrm{E}(x,y)$ の定義とみなすことにする．この $\mathrm{E}(x,y)$ は，対数誤差を無視すれば距離の公理を満たしている．

定義より $\mathrm{K}(x|y) \le \mathrm{K}(x)$ だが，たとえば，$\mathrm{K}(x|y)$ が小さいということは，y を知った上で，x を表わすのに必要なビット長が小さいということである．これは x が y に近いからである．一方，$\mathrm{K}(x|y)$ が大きく上界 $\mathrm{K}(x)$ に近い場合には，y という情報は x (の生成)に無縁であると考えられる．このように考えれば，$\mathrm{K}(x|y)$ を x の y に対する非類似度，すなわち x の y からの差とみるのは，直感的にも同意できるだろう．$\mathrm{E}(x,y)$ には対称性があるので，技術的には $\mathrm{K}(x|y)$ と $\mathrm{K}(y|x)$ の最大値を $\mathrm{E}(x,y)$ とするのである．

距離 $\mathrm{E}(x,y)$ は絶対量であるため，一般には，データの大きさ，つまり，列 x，y の長さに応じて大きくなる．正確には，x，y の本質的な複雑度 $\mathrm{K}(x)$，$\mathrm{K}(y)$ の大きさに依存する．そこで，大きさに依存しない相対的な類似度を議論するために，$\mathrm{E}(x,y)$ を次のように正規化した距離 $\mathrm{NID}(x,y)$ が提案されている．

$$\mathrm{NID}(x,y) \;=\; \frac{\max\{\mathrm{K}(x|y), \mathrm{K}(y|x)\}}{\max\{\mathrm{K}(x), \mathrm{K}(y)\}}. \tag{5.1}$$

この量は正規化情報距離 (normalized information distance) と呼ばれている．

Li et al. (2004)は，コルモゴロフ記述量に関するこれまでの研究を用いて，正規化情報距離の持つ様々な性質を明らかにした．とくに，正規化情報距離が，文字列間(データファイル間)の差異を表わす汎用の尺度として妥当なものであることが示されている．この距離を用いれば，すべてのデータファイル間の情報の類似度を統一的に測ることができるのである．

けれども，実際には正規化情報距離を用いることはできない．コルモゴロフ記述量が計算不可能だからである．そこで，コルモゴロフ記述量を計算できるもので代用することを考えてみよう．x を与えられた文字列とする．この x に対する記述量 $\mathrm{K}(x)$ は，x を究極に圧縮したときの圧縮列 p_x の長さと考えてよい．p_x から計算により x を復元することはできる．けれども，x から p_x へ圧縮する計算は，一般には不可能で，そのために $\mathrm{K}(x)$ が計算不可能なのである．そこで，何らかの圧縮アルゴリズム COMPRESS を考え，それが計算する圧縮列 $\mathtt{COMPRESS}(x)$ の長さを，$\mathrm{K}(x)$ の代用と考える．つまり，

$$C(x) \;=\; |\mathtt{COMPRESS}(x)|$$

を記述量 $\mathrm{K}(x)$ の近似と考えるのである．これを以降は擬似記述量と呼ぶことにしよう．

ここで考える圧縮は情報損失のない圧縮である．つまり，圧縮アルゴリズム COMPRESS の計算が，関数として 1 対 1 になっており，x の圧縮列 $\mathtt{COMPRESS}(x)$ から x が一意に復元できる圧縮である．Vitányi 等は，そのような圧縮を実現するアルゴリズムとして，通常良く使われている gzip や bzip2 などを用いて実験している．これらは高速ではあるが，一般には究極の圧縮からはほど遠い．また，2 つの文字列 x，y の並びを圧縮する場合には，xy と yx で圧縮率に大差がない方が望ましい．実際に，$\mathrm{K}(xy) \approx \mathrm{K}(yx)$ はつねに成り立つし，bzip2 も同様の性質を持つ．しかし，gzip では，そのような性質を保証できない．けれども Vitányi 等は，一連の実験では gzip と bzip2 は大差がなく，ほとんどの場合，どちらも同じ分類を導き出す，と報告している．以下では，擬似記述量 $C(x)$ の定義に用いる圧縮アルゴリズムには，たとえば bzip2 が用いられているものとして議論を進めよう．

擬似記述量 $C(x)$ を $\mathrm{K}(x)$ の代用に用いるには，もう 1 つ解決しなければならない点がある．式 (5.1) では，条件付き記述量(例：$\mathrm{K}(x|y)$)が用いられており，それを近似する方法が必要である．ここで，条件付き記述量に関する次の結果が重要な意味を持ってくる．

定理 6. （L.A. Levin（Li and Vitányi, 1993 参照））任意の列 x，y に対して次の関係が成り立つ．$\mathrm{K}(xy) \;\approx\; \mathrm{K}(x|y) + \mathrm{K}(y)$.

脇道にそれるが，この定理について少し述べておこう．

この近似等式のうち，$\mathrm{K}(xy) \lessapprox \mathrm{K}(x|y) + \mathrm{K}(y)$ は簡単に示せる．列 xy を生成するのに，y を生成するプログラムと，y を補助情報に用いて x を生成するプログラムがあればよい，という考え方から示すことができる．一方，逆の不等式の証明は技巧的である．また，逆の不等式が成り立つということは，列 xy を生成する方法として，上記の方法がほぼ最適であることを意味している．

また，定理の対称性と $\mathrm{K}(xy) \approx \mathrm{K}(yx)$ から，

$$\mathrm{K}(x) + \mathrm{K}(y|x) \;\approx\; \mathrm{K}(xy) \;\approx\; \mathrm{K}(y) + \mathrm{K}(x|y)$$

が成り立つ．したがって，

$$\mathrm{K}(x) - \mathrm{K}(x|y) \;\approx\; \mathrm{K}(y) - \mathrm{K}(y|x)$$

となる．この左辺は y に含まれる x の情報量，右辺は x に含まれる y の情報量を表わしていると考えることができる．そこで，この両者がほぼ等しいことは(そして定理 6 も)シャノンの相互情報量の対称性との類似性から情報の対称性 (symmetry of information) と呼ばれている．

さて，NID の近似の話に戻ろう．定理 6 の式は

$$\mathrm{K}(x|y) \;\approx\; \mathrm{K}(xy) - \mathrm{K}(y)$$

と書き換えられる．これを使うと式 (5.1) は次のように，条件付き記述量を使わないでも定義できる．

$$\mathrm{NID}(x,y) \;=\; \frac{\max\{\mathrm{K}(xy) - \mathrm{K}(x), \mathrm{K}(xy) - \mathrm{K}(y)\}}{\max\{\mathrm{K}(x), \mathrm{K}(y)\}}.$$

ここまで来れば，K を擬似記述量 C で置き換えることで，次の量 NCD が定義できる．これを x，y の差異を表わす簡易版情報距離として用いるのである．

$$\mathrm{NCD}(x,y) \;=\; \frac{\max\{C(xy) - C(x), C(xy) - C(y)\}}{\max\{C(x), C(y)\}}.$$

この簡易版情報距離は非常に使いやすい．2 つのファイルとそれをつなげたファイルの 3 つを bzip2 で圧縮し，そのビット長を求めれば NCD が計算できるのである．実際，Vitányi らは，遺伝子情報から文学作品や音楽ファイルまで，様々なファイル間の距離を測り，それに基づいて分類した結果を発表している．これらの結果を見ると，簡易版情報距離は汎用な類似度の計算法の有望な 1 つのように思われる．今のところ，なぜ bzip2 のような比較的表層的な文字ベースの圧縮アルゴリズムが与える長さを，真のコルモゴロフ記述量の代わりに利用できるのかが不明である．この点，今後のおもしろい研究課題になると期待している．

簡易版情報距離は，言われてみれば非常に自然な尺度である．けれどもコルモゴロフ記述量の専門家から提案されるまで誰も提案してこなかった．それはコルモゴロフ記述量の研究が「役立った」証拠と言っても過言でないだろう．

6. おわりに

ランダムネスや情報の類似度など，情報の基礎となる概念を「計算」という観点からながめた研究の一端を紹介した．本稿により，計算やプログラムが単なる道具ではなく，基礎概念を数学的に分析する研究手法の鍵となること，また，ランダムネスを明らかにする，という哲学的とも思える研究の中から，おもしろく有望な技術が生み出されていることを知って頂ければ幸いである．

今回は計算可能性の立場から観た研究について述べたが，計算時間やメモリ領域など，計算資源を限った上での計算から見ると，また異なる側面が現れてくることを最後に注意しておく．たとえば，多項式時間計算可能性の枠組みでは，RSA のような暗号システムが安全ならば，定理 6（情報の対称性）は成り立たないことが示されている．

参 考 文 献

Bennett, C., Gács, P., Li, M., Vitányi, P. and Zurek, W. (1998). Information distance, *IEEE Transactions on Information Theory*, **44**(4), 1407–1423.

Cilibrasi, R. (2003). The CompLearn Toolkit, http://complearn. soruceforge.net/.

Cilibrasi, R. and Vitányi, P. (2005). Clustering by compression, *IEEE Transactions on Information Theory*, **51**(4), 1523–1545.

Li, M. and Vitányi, P. (1990). Kolmogorov complexity and its applications, *Handbook of Theoretical Computer Science* (ed. J.van Leeuwen), 187–254, Elsevier, Amsterdam. (邦訳：Kolmogorov 記述量とその応用（渡辺 治 訳），『コンピュータ基礎理論ハンドブック』（廣瀬 健，小林孝次郎，野崎昭弘 監訳），187–262，丸善，東京）

Li, M. and Vitányi, P. (1993). *An Introduction to Kolmogorov Complexity and Its Applications*, Springer, New York.

Li, M., Chen, X., Ma, B. and Vitányi, P. (2004). The similarity metric, *IEEE Transactions on Information Theory*, **50**(12), 3250–3264.

Martin-Löf, P. (1966). The definition of random sequences, *Information and Control*, **9**, 602–619.

Proceedings of the Institute of Statistical Mathematics Vol. 54, No. 2, 511–523 (2006) 523

Randomness from Computational View Points

Osamu Watanabe

Department of Mathematical and Computing Sciences, Tokyo Institute of Technology

This paper surveys research for clarifying the notion of randomness from computational viewpoints. It first explains Martin-Löf randomness for infinite sequences. Then, after reviewing self-delimiting Kolmogorov complexity, it explains Kolmogorov randomness for finite sequences and its relation to Martin-Löf randomness. Finally, it explains recent applications of the Kolmogorov complexity notion to definition of univeral similarity distance.

Key words: Computability theory, Kolmogorov complexity, Martin-Löf randomness of infinite sequences, Kolmogorov randomness of finite sequences, information distance.