「理工学研究科総合講義 A」講義録
明治大学　2017 年度春学期　火 4 限

明治大学理工学部数学科　宮部賢志

2017 年 7 月 18 日

目次

1	第 1 回　乱数は様々な場面で使われる	3
2	第 2 回　乱数生成法：一様乱数	6
3	第 3 回　乱数生成法：一般の分布	9
4	第 4 回　乱数の検定：統計的検定	12
5	第 5 回　乱数の定義：計算可能性	15
6	第 6 回　弱い乱数：正規数	17
7	第 7 回　乱数の性質：極限定理	19
8	第 8 回　乱数の定義：有限列の情報量	23
9	第 9 回　乱数の応用：暗号	28
10	第 11 回　擬似乱数：計算複雑性クラス	32
11	第 12 回　擬似乱数の定義：計算複雑性クラス	34
12	第 13 回　科学哲学	38
13	第 14 回　科学におけるランダム性	40

前書き

　このノートは明治大学において 2017 年度春学期火 4 限に行う「理工学研究科総合講義 A」の講義録です．予習復習などに利用してください．

1　第 1 回　乱数は様々な場面で使われる

1.1　オリエンテーション

この講義の主題は「乱数」である．

「乱数」の理論は，計算，確率，情報など様々な概念および理論が関わる複合領域である．様々な分野の知識の組み合わせを学ぶことにより，理論の関係や使われ方を知ってほしい．

また「乱数」は様々な場面で使われるため，その使われ方も要求される性質も様々である．その場面や目的に応じて，乱数の概念の見え方がどのように変化するのかを明らかにすることもこの講義の目的の 1 つである．

この講義は大きく 4 つに分けられる．第 1 回から第 3 回までは，乱数の使い方という実務的な話をする．第 4 回から第 8 回までは，乱数の良さを統計的観点から考察する．第 9 回から第 11 回までは，乱数とは何かを計算論の立場から考察する．第 12 回から第 14 回までは，擬似乱数について計算量の立場から考察する．

1.2　乱数の使われ方 1: ゲーム

一番身近な乱数の使われ方はゲームかもしれない．例えば「すごろく」では，サイコロまたはルーレットなどそれに類するもので，次に進むコマ数を決める．通常のサイコロを使うとすれば，次に 1 から 6 までのどの目が出るか分からないことが，面白さの要因になっている．音楽におけるランダム再生もこのような例の 1 つだろう．

この場合，1 から 6 までの乱数を使っている．数の決め方はサイコロである必要はなく，次に出る目の予測が難しい数の列，であることが重要なのである．このような数を乱数と呼ぶ．

今，ここでは乱数の定義をしたわけではない．一般的に乱数という言葉はこのような意味で使われる，という説明をしただけである．この講義では，乱数という言葉の数学的定義を場面ごとに繰り返し行う．乱数という言葉を使った時，日常用語なのか，数学用語なのかの区別をしながら聞くことは重要である．

また例えば，アクションゲーム（プレイヤーを操作して，戦ったり，移動したりする．スーパーマリオブラザーズなど）において，プレイヤーの操作が全く同じならば敵なども全く同じ動きをするものもあれば，プレイヤーの操作が全く同じでも敵の動きが場合によって異なるものもある．どちらが良いか，面白いかは，もちろん場合による．適度な予測不可能性を持たせるために，乱数を使って，その都度異なる動きをさせることがある．例えば，乱数を使って，1 であればこのような動き，2 であればこのような動き，というような具合である．実際に数が現れなくても，内部では乱数が使われていることが多い．

2012 年にはコンプガチャが社会問題となった．ガチャとは，カプセルトイのことで，お金を入れるとカプセルに入ったおもちゃが出てきた．地域によって呼び名が違うらしいが，私の家の近くではガチャガチャと呼んでいた．どんなおもちゃが出てくるかは，お金を入れてカプセルを受け取り開けるまで分からない．すでに持っているおもちゃが出てくることもあるだろう．欲しいおもちゃが出るまでお金をつぎ込むように促されているのである．同じ仕組みが携帯のゲームで行われるようになった．しかもアイテムを全て集めると，つまりコンプリートすると，稀少アイテムが得られる．この仕組みをコンプガチャという．実際にやってみると，最初は勢いよく集まるが，全て集まるには多くの回数がかかる．「本当に公平に出ているのだろうか？」という疑問も出てくるだろう．計算機でガチャをシミュレートするには，計算機で乱数を作る必要がある．単に予測不可能というだけではなく，別の性質も要求されていることが分かる．

1.3　乱数の使われ方 2: ランダムに選ぶ

昔々，電話が 1 家に 1 台で家に固定されていて，誰か 1 人くらいは家にいるのが普通だった時代があった．インターネットはもちろん存在しない．政治の政策に対するアンケート調査では，よく家に電話をかけるという方法が取られた．もちろん日本全国全ての家に電話をかけるわけにはいかないので，適当に選ばなければならない．この選ぶときには偏りがあってはいけない．性別や年齢や仕事に偏りがあれば，調査の結果が日本の代表とは言えなくなる．そこで使われるのがランダムに選ぶという方法である．Random Digit Dialing(RDD) と呼ばれる．特定の規則なく

選んだのであれば，高い確率で日本の代表値と見なせるだろう，という考え方である．

「どうやって私が選ばれたのですか？」「ランダムです.」「そんないい加減なやり方は信用できません.」

このようなやりとりが実際にあるらしい．ランダムの有用さへの無理解が引き起こすすれ違いであろう．これが徴兵だったら，そういう気持ちになるのも無理はない．これが当選の電話だったら，疑ったほうが良いだろう．

ランダム利用の考え方を最初に提唱したのは現代統計学の父と言われるフィッシャー (1890-1962) であり，現在では，ランダム化比較実験と言われている．

> 何人かの英国紳士と夫人たちが屋外のテーブルで紅茶を楽しんでいたときのことだった．その場にいたある夫人はミルクティについて「紅茶を先に入れたミルクティ」か「ミルクを先に入れたミルクティ」か，味が全然違うからすぐにわかると言ったらしい．（中略）紳士たちのほとんどは，婦人の主張を笑い飛ばした．彼らが学んだ科学的知識に基づけば，紅茶とミルクが一度混ざってしまえば何ら化学的性質の違いなどない．だが，その場にいた 1 人の小柄で，ぶ厚い眼鏡をかけ髭を生やした男だけが，婦人の説明を面白がって「その命題をテストしてみようじゃないか」と提案したそうだ．

この人物がフィッシャーであり，最初のランダム化比較実験と言われている．“Milk In First(MIF)” と “Milk In After(MIA)” をランダムにした 10 つのミルクティーを作り，どちらなのかを当てさせれば良い．全て当てたのであれば，その確率は 2^{-10} であり，違いが分かると判定して良いだろう．ちなみにこの論争は 1870 年ごろから始まり，2003 年に the Royal Society of Chemistry が “How to make a Perfect Cup of Tea” を発行して決着がついている．結論は「ミルクが先」であり、乳成分に含まれるたんぱく質の変性が少なく、より美味しく仕上がるというものであった．

このランダム化比較実験は何らかの主張を行うのに必須の手法となっている．例えば，肥料 A と肥料 B のどちらが良いかを比較したいとする．ある年に肥料 A を使い，次の年に肥料 B を使って，収穫量を比較しても，どちらが良いのかは分からない．肥料以外に異なる要因が多くあるからである．同じ年に隣の畑で使ったとしても，同じである．奇数列は肥料 A，偶数列は肥料 B を使っても，その方向に養分が偏っている可能性もある．一方，ランダムに肥料 AB を使った場合には，肥料以外の要因が打ち消しあうので，純粋に肥料の差を調べることができる．

これで分かるように，何らかの主張をする上での公平性の担保にもランダム性が使われる．乱数が乱数であることが保障されていないと，その主張も危ういものになる．

1.4　乱数の使われ方 3: 確率モデルのシミュレーション

かつて乱数が必要な時には，乱数表を使った．ランダムに数が並んだ表である．しかし，今日では表に書ききれないほど多くの乱数が必要になることがある．

例えば，体内の器官の化学反応のシミュレーションを行いたいとしよう．それにより薬がどのように体に作用するのか，メカニズムを解明し，より良い薬の開発や副作用の低減につながるので，とても重要な問題である．分子の動きはランダムに見える．ニュートン力学的には決定的であってでも，あまりにも分子数が多いため，直接のシミュレーションは計算量が爆発して行えない．様々な未知のパラメータがあることもシミュレーションが行えない原因の 1 つである．そこで，統計力学的なシミュレーションを行う．ある一定の確率で反応するとモデル化するのである．

そのような確率モデルを計算機で行うときには問題が起こる．確率モデルを決定的な計算しかできない計算機にどのようにシミュレーションさせるか，という問題である．そこで確率モデルの標本として，乱数を使ってシミュレーションを行うのである．確率という概念と乱数という概念の関係は，一言で説明するのは難しい．しかし，乱数とは確率モデルの標本として不自然でないもの，という関係は成り立ちそうである．

現在は計算速度もメモリも格段に進歩したため，非常に複雑な現象のシミュレーションも行えるようになった．同時に膨大な量の乱数が必要になった．そのため，長期の列で見ても乱数に見える列を短時間で計算したいという欲求が出てきた．その列が乱数でなければ，シミュレーションの結果も保障できない．

1.5　乱数の使われ方 4: 乱拓アルゴリズム

乱数が必要になるのは確率モデルだけではない．インターネット上で，ある計算機と別の計算機を最短で結ぶ回線を探す問題を考えよう．これは，グラフが与えられたときに，ある点からある点まで最短距離で進む道を見つけるという問題に帰着できる．真の最短距離を見つけるためには，全探索すれば良く，そのような計算は原理的には可能である．しかし，グラフが大きい場合には，計算量は爆発的に大きくなり，実用的な時間では計算できなくなる．

ところが，ランダムに点を選んで，うまく計算すると，かなり高い確率で最短距離に近い道を見つけるようなアルゴリズムを考えることもできる．このような乱数を使うことで，短い時間で計算ができるようなアルゴリズムを，乱拓アルゴリズムという．

別の有名な別の例として，ミラーラビン素数判定法などがある．

1.6　乱数の使われ方 5: 暗号

古来から人類は戦争をしてきた．戦争の戦略を伝えるために暗号文が発達した．暗号文はある人には読むことができ，別の人には読むことができない，という条件を満たす必要がある．

もっとも原始的な暗号として，乱数表を使うものがある．a から z までの文字で文章が書かれているとする．乱数表の順番に，a から z の文字をずらしたものを，暗号文として送る．乱数表を持っている人は読めるが，持っていない人は読めない．もし，その乱数表に分かりやすい規則があれば，乱数表を持っていなくても読めてしまうかもしれない．

現代のインターネット上では暗号はとても重要な技術である．https の s は secure を意味し，暗号化して通信する．インターネットで買い物をするときには，情報を暗号化しないと，クレジットカードの情報を誰でも見れることになってしまう．現在の主流の暗号方式は RSA 暗号であり，素因数分解の困難性に依拠している．しかし RSA 暗号そのものでは十分な安全性は保てないため，乱数を使って安全性を確保する手法が使われている．

もっと単純な例として，パスワードの生成が挙げられる．単純なパスワードでは総当たりで破られるので，複雑なものにする必要があるが，それを乱数で生成することがある．ネットバンキングでは，つい最近まで乱数表を使っていた．

良い乱数は，私たちの安全の根拠となるものでもある．

2　第 2 回　乱数生成法：一様乱数

2.1　乱数の種類

最も原始的な乱数はアストラガルスと言われる．アストラガルスは動物の骨で作ったサイコロで，人類と同じくらいの歴史があり，起源はハッキリしない．占いやゲームとして使われていたようである．ファイナルファンタジー XI やサプリメントにも同じ名前のものがでてくるらしいが，関係は私は知らない．乱数を作るためのサイコロを乱数賽という．しかし 1 つの乱数を作るためにサイコロを投げると言うのはあまりにも手間がかかりすぎる．

科学における乱数の重要性は統計で最初に気がつかれたようである．19 世紀終わり頃から，品質管理の目的で，製品をランダムに選んで検査する，といったことが行われるようになった．Pearson の弟子である Tippett が 1927 年に乱数表を出版している．今でも，統計数理研究所のホームページから乱数データをダウンロードできる．乱数表はごく最近まで使われてきたが，大量の乱数が必要になると，乱数を保存するだけでも大変なことになる．

真の乱数として信頼できるものとして，物理乱数がある．熱雑音や原子核の分裂などランダムな自然現象を測定することで乱数を得る．時々しか呼ばれないと分かっているならば，CPU 時間の 1 桁目と言うのも楽である．しかし，現実には測定が大変なため，あまり使われない．上記の統計数理研究所のホームページからは物理乱数もダウンロードできる．

計算機が誕生した 1940 年代には，すでに計算に乱数を利用すると言うモンテカルロ法の考え方が現れていた．そこで計算機によって乱数を生成する方法が模索され始める．計算機で作るのである意味では予測可能であり真の乱数ではないため，擬似乱数と呼ばれる．

> Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin. (von Neumann)

それでも使う目的によっては十分な精度が得られることがある．何より短時間に大量に乱数が欲しい場合には現在考えられる唯一の手法である．

2.2　平方採中法 (middle-square method)

1946 年頃，von Neumann によって考案された方法である．例えば 6 桁の乱数が欲しい場合に，2 乗して得られた数の中央 6 桁を取り出すと言う方法である．

$x_0 = 0.753106$ ならば，$x_0^2 = 0.567168647236$ であり，$x_1 = 168647$ となる．同様にして，$x_2 = 0.441810$, $x_3 = 0.196076$ となる．しかし，数学的な解析が難しいことや，初期値によっては乱数には見えない数列となることから，あまり使用されていない．

2.3　乗算型合同法 (Multiplicative Congruence method)

レーマー (Lehmar) が 1949 年に考案した方法である．例えば，

$$\begin{aligned} x_{n+1} \equiv & 15 x_n \mod 10^6 + 1 \\ x_0 = & 1 \end{aligned}$$

として定義すると，

$$1, 15, 225, 3375, 759375, 390614, \cdots$$

となる．最初のしばらくを捨てて，759375 から始めれば，乱数に見える列を作ることができる．

一般に，

$$\begin{aligned} x_{n+1} \equiv & a x_n \mod m \\ x_0 = & b \end{aligned}$$

と定義できる．この方法の良いところは，a, b, m を適切に取れば，周期が計算できることである．

定理 2.1 (フェルマーの小定理)**.** p を素数とし，a と p は互いに素であるとすると，

$$a^{p-1} \equiv 1 \mod p$$

例えば，$a = 3, p = 7$ の場合，

$$a^1 \equiv 3,\ a^2 \equiv 2,\ a^3 \equiv 6,\ a^4 \equiv 4,\ a^5 \equiv 5,\ a^6 \equiv 1$$

より周期 6 の列となり，$1 \sim 6$ が 1 度ずつ現れ，ランダムに見える．

証明. $\{a, 2a, 3a, \cdots, (p-1)a\}$ を p を法として考える．これらは全て p の倍数ではないから，0 ではない．$ia \equiv ja \mod p$ とすると，$i \equiv j \mod p$ となるので，これらは全て p を法として異なる．すなわち $\{1, 2, \cdots, p-1\}$ に法として等しい．全て掛け合わせると，$(p-1)! \equiv (p-1)! a^{p-1}$. $(p-1)!$ と p は互いに素だから，$a^{p-1} \equiv 1$. □

2.4 線形合同法 (linear congruential method)

前述の方法を少し発展させて長く使われてきた方法である．

$$\begin{aligned} x_{n+1} &\equiv a x_n + c \mod m \\ x_0 &= b \end{aligned}$$

により定める．うまく a, b, c, m を定めれば，最大周期 m が実現できる．

保存領域が小さいことや，それなりに早いことが長所として挙げられる．ただし，組で使ったり，下の方の桁だけを使うと，規則が見つかることが分かっている．

70-90 年代の C 言語標準の乱数生成器．$a = 1103515245,\ c = 12345,\ m = 2^{31}$ で周期が m.

2.5 メルセンヌツイスター (Mersenne twister)

1996 年に発表された松本眞と西村拓士による方法.

$$x_{n+p} = x_{n+q} + x_{n+1} B + x_n A$$

ここで，x_n は w ビットのベクトルで，A, B は $w \times w$ 行列.

MT19937 と呼ばれる生成法では，周期 $2^{19937} - 1$ を持つことが示されている．長所は周期が長いこと．短所は 19937bit(約 2.5KB) のワーキングメモリが必要なこと．

Python, Ruby, R, PHP, MATLAB などの標準ライブラリになっている．

2.6 Xorshift

2003 年の Marsaglia により発表された方法．2015 年 12 月 17 日に Chrome の Javascript での Math.random() 関数のアルゴリズムとして，xorshift+ を採用することが発表された．

y を例えば 32bit 整数として，

```
y^=y<<13; y^=y>>17; y^=y<<15;
```

ここで，^は xor を，>>,<<はそれぞれ右シフト，左シフトを表す．

y を 32 行のバイナリベクトルとすると，xor は足し算を表し，ビットシフトは

$$L_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

の積で書けるので，上記の漸化式は，

$$y = (I + L_{13})(I + R_{17})(I + L_{15})y$$

と書ける．

今，零ベクトル以外の $2^{32} - 1$ 種類を全て取るためには，行列の位数を調べれば良い．少し工夫をすれば，全ての場合を書き出すのに数分で済む．論文には 81 通り存在すると書かれている．$(a, b, c) = (13, 17, 15)$ はそのうちの 1 つ．

3　第 3 回　乱数生成法：一般の分布

3.1　離散分布

一様分布の乱数が与えられている時に，離散的な分布を得るアルゴリズムを与えよう．

1 から 6 までが均等に出る乱数を作りたいとする．$[0,1]$ までの一様乱数があるならば，6 倍してその整数部分を取れば良い．

ではポアソン分布に従う乱数を生成するにはどうすれば良いだろうか．ポアソン分布とは，定数 $\lambda > 0$ に対し，

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!},\ k = 0, 1, 2, \cdots$$

で表される確率分布である．平均と分散はそれぞれ $E(X) = \lambda$, $V(X) = \lambda$ である．大雑把に言って単位時間当たりの現象の発生回数を表す．1 日に受け取るメールの件数，1 分間の web サーバへのアクセス数，1 時間当たりのウィキペディアの更新ページ数，1 日の客の数，など．

ポアソン分布は 2 項分布の極限として表される．単位時間を n 分割する．各分割時間において確率 p で現象が発生するとする．単位時間で起こる回数の期待値は np である．そこで $\lambda = np$ を保ったまま，$n \to \infty$ とすると，

$$\begin{aligned} P(X = k) = \binom{n}{k} p^k (1-p)^{n-k} =& \frac{n!}{k!(n-k)!} \frac{\lambda^k}{n^k} \left(1 - \frac{\lambda}{n}\right)^{n-k} \\ =& \frac{n(n-1)\cdots(n-k+1)}{n^k} \frac{\lambda^k}{k!} \left(1 - \frac{\lambda}{n}\right)^{n} \left(1 - \frac{\lambda}{n}\right)^{-k} \\ \to& \frac{\lambda^k e^{-\lambda}}{k!}. \end{aligned}$$

よって，もっとも単純なシミュレーションとしては，十分大きな数 n で時間を分割し，発生確率を $p = \lambda/n$ として，p より小さければ現象が発生し，p より大きければ現象が起こらなかったとすれば良い．この方法では単に起こった回数だけを知りたい場合には，効率がとても悪い．

$P(X = k)$ を $[0,1]$ 区間にそれぞれ割りあてる．$u \in [0,1]$ が一様乱数として与えられたら，$\sum_{k=0}^{x-1} P(X = k) < u \leq \sum_{k=0}^{x} P(X = k)$ を満たす x を返す．このようなことが起こる確率は $P(X = x)$ であることに注意せよ．等号は確率 0 のことなので，気にしなくて良い．

3.2　連続分布：逆関数法

連続の確率分布の乱数を得るアルゴリズムを与えよう．

確率変数 X の密度関数とは，

$$P(a \leq X \leq b) = \int_a^b f(x)dx$$

を満たす関数 f のことであった．累積分布関数は，

$$F(x) = P(X \leq x)$$

として定義される．

指数分布とは，正のパラメータ λ に対し，密度関数が，

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & (x \geq 0) \\ 0 & (x < 0) \end{cases}$$

で与えられる分布であった．ポアソン分布が単位時間当たりの生起回数であったのに対し，指数分布は次に起こるまでの時間である．累積分布関数は，

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & (x \geq 0) \\ 0 & (x < 0) \end{cases}$$

で与えられ，期待値と分散は $E(X) = \frac{1}{\lambda}, V(X) = \frac{1}{\lambda^2}$ となる．

指数分布の乱数を得るには，逆関数法を用いると良い．

定理 3.1. F を確率変数 X の累積分布関数とする．U が $[0,1]$ の一様分布に従う時，$F^{-1}(U)$ は X と同じ分布に従う．

証明. $u \in [0,1]$ として，

$$P(U \leq u) = u$$
$$P(F^{-1}(U) \leq F^{-1}(u)) = u$$
$$P(F^{-1}(U) \leq x) = F(x)$$

これは，確率変数 $F^{-1}(U)$ の累積分布関数が F であることを意味する． □

指数分布の場合，$F(x) = 1 - e^{-\lambda x}$, $(x \geq 0)$ であるから，

$$F^{-1}(x) = -\frac{1}{\lambda}\log(1-x)$$

x が一様分布に従えば，$1-x$ の一様分布に従うので，結局，x を一様乱数として，$-\frac{1}{\lambda}\log x$ を計算すれば良い．

3.3 正規分布

正規分布は，平均を μ, 分散を σ^2 として，密度関数が

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

で表されるものである．これを $N(\mu, \sigma^2)$ と書く．

正規分布は誤差の分布として現れることが多い．身長，体重，ノイズ，など．

統計において正規分布が重要なのは，中心極限定理が成り立つからである．期待値 μ，分散 σ^2 の独立同分布の確率変数列 $X_1, X_2, \cdots$ に対し，$S_n = \sum_{k=1}^{n} X_k$ とおくと，

$$P(\frac{S_n - n\mu}{\sqrt{n}\sigma} \leq \alpha) \to \int_{-\infty}^{\alpha} \frac{1}{\sqrt{2\pi}}\exp(-\frac{x^2}{2})dx$$

すなわち，標準正規分布に収束する．

$[0,1]$ の一様分布の期待値は $\frac{1}{2}$, 分散は $\frac{1}{12}$ であることから，一様分布を 12 個足して，6 を引けば，近似的に標準正規分布になる．場合によってはこれで十分な乱数となる．

正規乱数には直接は逆関数法は使えない．累積分布関数の陽な形が求まらないからである．よく使われるのが Box-Muller 法 (極座標法) である．

X, Y を独立な標準正規分布に従う確率変数とする．

$$\begin{aligned} P(a \leq X \leq b,\ c \leq Y \leq d) =& P(a \leq X \leq b)P(c \leq Y \leq d) \\ =& \int_a^b \frac{1}{\sqrt{2\pi}}\exp(-x^2/2)dx \int_c^d \frac{1}{\sqrt{2\pi}}\exp(-y^2/2)dy \\ =& \int_a^b \int_c^d \frac{1}{2\pi}\exp(-(x^2+y^2)/2)dxdy \end{aligned}$$

であることに注意する．$X = R\cos\Theta$, $Y = R\sin\Theta$ となる確率変数 R, Θ を考える．$R = \sqrt{X^2+Y^2}$ であるから，R の累積分布関数は，

$$\begin{aligned} P(R \leq \alpha) =& \iint_D \frac{1}{2\pi}\exp(-(x^2+y^2)/2)dxdy \qquad D = \{(x,y) \mid x^2+y^2 \leq \alpha\} \\ =& \int_0^{\alpha}\int_0^{2\pi} \frac{1}{2\pi}\exp(-r^2/2)r\ drd\theta \\ =& 1 - \exp(-\alpha^2/2) \end{aligned}$$

と陽に書ける．この逆関数は，$y = \sqrt{-2\log(1-x)}$ であるから，u, v を $[0,1]$ の一様乱数として，

$$r = \sqrt{-2\log u},\ \theta = 2\pi v,\ x = r\cos\theta,\ y = r\sin\theta$$

とすれば，x, y は標準正規分布に従う乱数となる．

$\cos, \sin$ の計算が重い場合には，以下のような方法もある．U, V を $[0,1]$ 上一様分布する確率変数として，$S = U^2 + V^2$ という確率変数を考えると，その累積分布関数は，$x \in [0,1]$ において，

$$P(S \le x) = P(U^2 + V^2 \le x) = \int_0^{\sqrt{x}} \int_0^{\pi/4} r\ drd\theta = \frac{\pi}{8}x$$

なので，一様．そこで，u, v を $[0,1]$ の一様乱数とし，$s = u^2 + v^2$ を計算して，$s \in (0,1)$ 以外の場合は棄却する．そして，

$$x = \sqrt{-2\log s}\frac{u}{\sqrt{s}},\ y = \sqrt{-2\log s}\frac{v}{\sqrt{s}}$$

とすれば，x, y は標準正規分布に従う乱数となる．

4　第 4 回　乱数の検定：統計的検定

4.1　乱数に求められる性質

これまで乱数の使い方と擬似乱数の作り方について話してきた．そもそも乱数とは何だろうか．

1 つだけの数が乱数であると言われても，検証のしようがない．乱数と呼ばれるのは数の列である．その数の列の規則が見つけられず，予測不可能である様をいう．一方で，整数であるとか，少数であるとか，32bit であるなどの条件は分かっていても良いので，何かしらの規則は存在する．そこで何かしらの規則の存在は前提にした上で，それ以上の規則が見つけられないことをいう．多くの場合，何らかの確率変数の実現系列と見分けがつかないことを意味する．この乱数として最も必要とされる性質は，信頼性とも言われる．乱数の使用を正当化する性質であるからである．

モンテカルロ法など乱数を使ってシミュレーションを行う場合，再現性が求められる．実験科学の世界では，「このような手順で行うとこのような結果が出る」という報告が必要である．「たまたまできたが，再現できない」では困る．計算機によるシミュレーションは一種の実験である．そこで数値シミュレーションも再現できるように条件を整えるのが良い．乱数を使わない通常の数値計算でも浮動小数点の丸め誤差の影響などにより，同じプログラムでも環境によって答えが異なることがある．そこで計算機援用証明などでは丸めの方向などの環境の含めて論文に書かれることがある．乱数を用いたシミュレーションにおいては，乱数を使っているのであるから毎回答えが異なって当然である一方，再現性も確保したい．そこで種 (seed) が使われる．種はそこから乱数の初期値を作る．種が同じであれば同じ乱数が作られる．通常，プログラムで乱数を使いたい場合には，最初に種を初期化する．種さえ保存しておけば，同じ乱数が作られるので，再現性も確保される．

昨今の数値シミュレーションでは，分子レベルのシミュレーションなど，膨大なデータ数が必要になることが多い．例えば，ミリ秒単位で 10^5 個の分子をランダムに動かせば，毎秒 10^8 個の乱数が必要になる．このような状況では迅速性が重要となる．従来は加算，乗算の数が少ないほうが良いと言われていたが，最近では xorshift のように論理演算の数を減らすというレベルの話になってきている．

良い擬似乱数は，信頼性，再現性，迅速性などを満たすものである．後者の 2 つを確認するのは容易であるが，信頼性の検証の仕方には工夫が必要である．

4.2　擬似乱数検証ツール

(1) TestU01
(2) PractRand
(3) Dieharder
(4) NIST Special Publication 800-22

いずれも，テスト群のパッケージ．

4.3　χ^2 検定の実験

今日は手を動かしてみよう．

(1) 1 から 6 までの数字をそれぞれ確率 1/6 となるように，120 個書いてください．
(2) 1 から 6 までの個数を数えてください．
(3) それぞれから 20 を引いて二乗して，すべてを加えて，20 で割って，χ^2 値を計算してください．
(4) その値が，0.831212 から 12.8325 の間にあることを確認してください．

これが χ^2 検定もしくはピアソンの適合度検定と言われるものです．有意水準は両側 5% で計算しています．

一体何をやっているのか，もう少し見てみましょう．1 から 6 までの数字が確率 1/6 で表れる確率分布を考える．それを 120 個独立に試行すれば，それぞれ表れる期待値は 20 のはずである．実際にやってみると，それぞれ 20 から少しずれる．どれくらいのずれならば，不自然ではないと言えるだろうか，というのを問題にする．もし確率が 1/6

ずつでなければ，ずれは大きくなるだろう．ランダムでなければ，例えば 1,2,3,4,5,6 を順番にとるようであれば，ずれは小さすぎるということになる．確率 1/6 で独立であると信じても不自然ではないといえるのは，ずれがどれくらいの時だろうか？それは確率 1/6 で独立である場合に，そのずれがどのような確率分布を取るかを考えることで，その端にならなければ，不自然ではないと考えよう，というのが統計的検定の考え方である．

いくつか言葉を紹介しよう．最初に考える確率分布を帰無仮説と呼ぶ．この確率分布の標本点として不自然ではないかどうかを検証する確率分布のことである．何を主張したいかは場合によるが，「この確率分布では不自然である」ということが言いたい場合が多いので，最後にはその仮説は棄却されることを望んでいる．そのため，無に帰す仮説と呼ばれている．

その確率分布の上で，適当な値がどんな分布をとるかを考える．ここには高度な数学が使われることが多く，先人たちが求めたものを使うことが多い．最近では計算機で直接計算して求めるという方法がとられることもある．

次に有意水準を設定する．簡単に言えば，どのくらいの確率ならば，間違えても良いか，という基準である．間違え方には 2 種類ある．本当は正しいのに不自然だと判断する間違いと，本当は間違っているのに不自然ではないと判断する間違いである．最初の間違いの確率としてどの程度を許すかを表すのが有意水準である．2 番目の間違いの確率は，単純には計算出来ない．どちらも小さくするためには，試行回数を増やす必要がある．

この有意水準は最初に設定する必要がある．計算してからでは，検定にならない．具体的な列を見てから検定方法を変えてはいけないということである．

有意水準の設定の仕方としては，5%，1%，0.5% などが使われることが多い．どれが適当であるかは文脈によるが，5% を使うことが多い．この選び方は恣意的ではあるが，他の数字にするのはもっと恣意的になってしまう．

次にその有意水準にしたがって，端の部分は不自然だと判断することにする．なぜ，端なのか．独立性を仮定すれば，数値が違うと分布は端に動くので，この考え方は多くの場合において，不自然すぎるわけではない．しかし，やはり恣意的ではある．

端に入っていれば，仮説を棄却するという．仮説が不自然であるということを意味する．そうでなければ，仮説は採用する．仮説が不自然であるとはこの検定によっては言えないというだけなのだが，通常そのような表現がとられる．

今回の場合，6 個の数字の和である χ^2 値は，自由度 5 の χ^2 分布に従うことがわかっていて，その両側 5% 検定の数値が，例の値である．

4.4　様々な検定

実際の検定の現場で使われているものの中から，面白いものを 2 つ紹介する．

4.4.1　ポーカー検定

0 から $d-1$ の中から値を取る n 個の乱数列に対し，

(1) 5 つずつの組に分ける
(2) それぞれについて，1 種類，2 種類，3 種類，4 種類，5 種類の数を数える
(3) 適切な確率に対して，χ^2 を計算して，自由度 4 のカイ二乗分布で検定する．

4.4.2　誕生日間隔検定

0 または 1 の値を取る乱数列に対し，

(1) 32 ビットずつ区切る
(2) 各 32 ビットから 24 ビット取り出す
(3) 前ステップを 1024 回繰り返し，小さい順に並べてその 1023 個の間隔の値 $\{Y_i\}$ を求める
(4) Y_i を数直線にプロットして，重なった回数 s を数える
(5) 前 3 ステップを 500 回繰り返し，500 個の s を求める．s は $\lambda = (2^{10})^3/(4\cdot 2^{24}) = 16$ のポアソン分布に従うので，p 値を計算する

(6) 前 4 ステップを 9 回繰り返す．24 ビットを取り出す位置を変える

(7) 9 個の p 値が一様分布しているかどうか KS 検定を行う

5　第 5 回　乱数の定義：計算可能性

5.1　乱数の定義再訪

乱数とは「適当な確率分布の標本点として不自然でないもの」として定義した．そして乱数として自然かどうかは「多くの検定に合格するどうか」で調べると話した．それでは「すべての検定に合格する列」として乱数を定義できるのではなかろうか．実は，万能検定ともいうべき検定が圧縮による検定である．圧縮の概念を説明するために，計算の概念が必要になる．

今回は以下の 2 文について理解し，証明のスケッチを与えることである．

(1) 任意の計算可能関数を模倣する万能機械 (universal machine) が存在する．
(2) 任意の圧縮プログラム以上に圧縮できるプログラムが存在する．

ここで考えるのは，$f : 2^{<\omega} \to 2^{<\omega}$ の計算可能性である．素朴に考えて，自然数が与えられて，それを 2 倍するとか，素数かどうかを判定するなどは計算できる．その関数を計算するようなアルゴリズムが存在するということである．大雑把には，それを計算するようなプログラムが存在すると思って良い．

また，自然数と 2 進無限列には，

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \cdots$$
$$1, 10, 11, 100, 101, 110, 111, 1000, \cdots$$
$$\lambda, 0, 1, 00, 01, 10, 11, 000, \cdots$$

のような対応関係を作れば，1 対 1 の対応関係が作れる．これを符号化という．そこで，符号化した状態での計算可能性を考える．

計算可能な関数には，それを計算するアルゴリズムが存在するが，アルゴリズムもまた符号化できる．そこで，「符号化されたアルゴリズムと文字列を受け取って，そのアルゴリズムにその文字列を入力した時の結果を出力するアルゴリズム」を考えることができる．そのようなアルゴリズムが存在することは素朴には納得できるであろう．

実際にはアルゴリズムと文字列の結合の符号化には少し注意する必要がある．例えば，アルゴリズムの符号化が 110110011 で，文字列の符号化が 110 であったとする．そのままつなげれば，110110011110 となるが，この文字列を見て，どこまでがアルゴリズムの符号化で，どこからが入力文字列の符号化なのかが分からない．通常のプログラムでは，EOF や*などの終端記号を使って区切りを表す．しかし，今考えているのは 2 進無限列上の関数なので，これはできない．そこで，アルゴリズムの符号化の方を 2 重にして，最後に 01 をつける．今回の例で言えば，11110011110000111101110 などとすることで，復号できるようにしておく．

以上のことをまとめると，次のことが分かる．

定理 5.1 (Turing)**.** 以下の性質を満たす計算可能関数 $U :\subseteq 2^{<\omega} \to 2^{<\omega}$ が存在する．これを万能機械 *(universal machine)* と呼ぶ．任意の計算可能関数 $f :\subseteq 2^{<\omega} \to 2^{<\omega}$ に対して，ある文字列 $\sigma \in 2^{<\omega}$ が存在して，任意の $\tau \in 2^{<\omega}$ に対して，

$$U(\sigma\tau) = f(\tau)$$

が成り立つ．

$U :\subseteq 2^{<\omega} \to 2^{<\omega}$ は部分関数であることを表している．上記の定理は U が任意の計算可能関数 f を模倣できることを意味している．

通常のコンピュータは万能機械であり，インターネットからアプリをダウンロードすることで，もしくはプログラムを書くことで，どんな計算可能関数も計算できる．その意味ではこの存在定理は驚くことではないかもしれない．しかし，歴史的には逆で，この定理が発見された時には今日のようなコンピュータは存在しなかった．

今日のコンピュータはフォン・ノイマン型コンピュータとかプログラム内蔵型と言われる．プログラムをインストールすれば，どんな計算もできるようになるからである．そのような計算機の開発は 1940 年代に行われた．それ

以前にも計算機は存在したが，特定の機能しか持たず，特定の計算しかできないものである．今日でも電卓はそのような計算機と言えるだろう．そういう自体に Turing が万能機械の存在を証明し，その実現物という形でフォン・ノイマンらが計算機を作ることになる．

万能計算機があれば，世の中の仕組みが大きく変わるとか，お金が儲かるとか，そういうことを考えて万能機械というものを考えたのではない．その当時数学界で問題になっていた，ディオファントス方程式の決定問題の否定的解決に向けて，計算もしくはアルゴリズムの数学的定義を与える必要があり，Turing はその一翼を担ったのである．

6　第 6 回　弱い乱数：正規数

すでに述べたように「すべての計算可能な検定に合格する列」として乱数を定義したい．統計的検定では有限列のランダム性を検定したのに対し，これから定義するのは無限列のランダム性の検定である．そのわかりやすい例として正規数という概念があるので，今日はその紹介をする．

6.1　正規数の定義

$\frac{1}{7}$ を少数展開すると，

$$\frac{1}{7} = 0.\dot{1}42857\dot{}$$

となって循環する．少数展開が有限小数となることと，$\frac{c}{2^a 5^b}$ のように分数表示の分母の素因数が $2, 5$ だけであることは同値である．小数展開が循環することと，有理数であること，すなわち分数で表示できることは同値である．無理数の実数は少数展開すると循環しない．

$\frac{1}{7}$ の少数表示には $0, 3, 6, 9$ が出てこない．少数展開に 0 から 9 までが一様に出てくるような数を考えよう．式で書くと，$N(w, n)$ を最初の n 桁に w が表れる回数とすると，

$$\lim_n \frac{N(w,n)}{n} = \frac{1}{10}$$

を満たす数である．w を 2 桁以上でも良いとして，10 進数だけでなく一般の b 進数でも良いとすれば，

$$\lim_n \frac{N(w,n)}{n} = \frac{1}{b^{|w|}}$$

となる．これがすべての b と w に対して成り立つ数を，**絶対正規数** (absolutelly normal number) と呼ぶ．10 進だけならば 10 進正規数と呼ぶ．

絶対正規数の概念は，1909 年 Borel により導入された．Borel が示したことは「正規数でない数はルベーグ測度 0」である．すなわち，ほとんどすべての実数が正規数であり，確率 1 で正規数である．normal の名前のつく概念は多いので，Borel normal とも呼ばれる．

6.2　正規数の例

正規数を具体的に構成したのは 1933 年 Champernowne で，

$$0.12345678910111213 14\cdots$$

のように小さい順に数を並べたものが 10 進正規数であることを示した．他の基数では未解決である．

Copeland-Erdös は 1946 年に

$$0.2357111317192329313741 43\cdots$$

のように素数を順に並べたものが 10 進正規数であることを示した．

絶対正規数の数に移ろう．1937 年 Turing は計算可能な正規数が存在することを示した．Turing は Champernowne と友人であったらしく，Champernowne からこの問題を聞いたらしい．この時代以前には「計算可能」という概念の定義がなかったので，計算可能な数という定義を作った Turing が示すことができたのも不思議ではない．実際，この構成は Borel による正規数が測度 1 で存在することの証明を観察することによって得られる．

2012 年は Turing year であった．Turing 生誕 100 周年としてイギリスでは特に盛大なイベントが行われた．Computability in Europe という計算可能性理論の国際研究集会で，Verónica Becher が”Turing's Normal Numbers: Towards Randomness”というタイトルで講演を行った．多くの研究者が聞いていて，正規数に注目が集まった．なぜなら，ここ 10 年近くで急激に発展したランダムネスの理論がそのまま適用できるトピックであったから．

2013 年には多項式時間で計算可能な正規数の構成が，Lutz-Mayordomo, Figueira-Nies, Becher-Heiber-Slaman の 3 つのグループにより独立に示された．この後，計算可能性理論の研究者と代数学の研究者による正規数関連の研究集会も開かれるようになった．

すべての正規数は超越数であると信じられているが，これも未解決問題である．$\sqrt{2}, e, \pi, \log 2$ なども正規数であると信じられているが，これも未解決である．

6.3　情報の圧縮ができるか？

「多項式時間計算可能な正規数が存在する」ことが分かった．例えば，π などがその候補である．正規数はどんな有限列も含む．どんな情報をも含むと言っても良い．私の誕生日も含まれる．みなさんの将来の結婚相手の名前もコーディングされている．原子爆弾の作り方のレシピも，国家機密でも入っている．

ならば π の何桁目から何桁といえば情報を伝えられるはずだ．これでもう「ダウンロードが遅い」などと嘆く必要はない．数字 2 つを伝えるだけで，どんな情報でも伝えられるのだから．携帯電話の通信の大幅な高速化が期待できる？長距離の通信が安定する？

そういうわけにはいかない．2 進で n 桁を伝えるためには，その出現確率は 2^{-n} くらいなので，最初に出てくる桁の期待値は 2^n くらいとなる．いくら多項式時間計算可能でも 2^n 桁を計算するには指数時間かかる．さらに，2^n くらいの数を表現するには，$n + \log n$ 桁くらいかかることも分かる．まったく圧縮できていないことが分かる．

6.4　次回予告

正規数は検定に合格した数と見ることができる．だから正規数は弱い乱数である．計算可能な検定は他にも存在する．計算可能な検定はどのように表現したら良いのか．まずは Borel 正規数が測度 1 で存在することの特殊例である大数の法則を示して，確率 1 であるという概念に慣れよう．

7　第 7 回　乱数の性質：極限定理

乱数の定義を目標にしている．乱数はすべての計算可能な検定を通る列として定義したい．正規数は弱いランダムな列の例であることを説明した．今日は正規数の条件が検定とみなせることを説明する．

7.1　Chebyshev の不等式

$X_1, X_2, \cdots$ を $P(X_i = 1) = p, P(X_i = 0) = 1 - p$ となる独立同分布の確率変数列とする．$S_n = \sum_{i=1}^{n} X_i$ とおく．

確率変数 X の期待値 $E(X)$ は，

$$E(X) = \sum_{k=1}^{\infty} x_k P(X = x_k)$$

として定義される．また，分散 $V(X)$ は，

$$V(X) = E((X - E(X))^2) = \sum_{k=1}^{\infty} (x_k - E(X))^2 P(X = x_k)$$

として定義される．標準偏差は，

$$\sigma(X) = \sqrt{V(X)}$$

として定義される．期待値はどの程度の値をとるか，分散はどれくらい散らばっているかを表す値である．

S_n がどんな分布をするのか調べてみよう．まず，期待値は，

$$E(S_n) = E(\sum_{i=1}^{n} X_i) = \sum_{i=1}^{n} E(X_i) = np.$$

分散は，

$$\begin{aligned} V(X_i) =& (1-p)^2 p + (0-p)^2 (1-p) = p(1-p), \\ V(S_n) =& V(\sum_{i=1}^{n} X_i) = \sum_{i=1}^{n} V(X_i) = np(1-p) \end{aligned}$$

と計算できる．

値は期待値の周辺に集まることが多いので，端の確率は小さいはずである．

定理 7.1 (Chebyshev の不等式)**.**

$$P(|X - E(X)| \geq k\sigma(X)) \leq \frac{1}{k^2}$$

証明.

$$D = \{\omega \ : \ |X - E(X)| \geq k\sigma(X)\}$$

とおくと，

$$V(X) = \sum (x_k - E(X))^2 P(|x_k - E(X)| \geq k\sigma(X)) = k^2 V(X) P(D)$$

より，$P(D) \leq \frac{1}{k^2}$. □

定理 7.2 (大数の弱法則)**.**

$$P(|S_n/n - p| \leq \epsilon) > 1 - \frac{p(1-p)}{\epsilon^2 n}$$

証明.

$$P(|S_n/n - p| > \epsilon) = P(|S_n - np| > \epsilon n) = P(|S_n - np| > \frac{\epsilon\sqrt{n}}{\sqrt{p(1-p)}} \sqrt{np(1-p)}) \leq \frac{p(1-p)}{\epsilon^2 n}$$

□

定理 7.3 (大数の強法則)**.**

$$P(\lim_n S_n/n = p) = 1.$$

更に，計算可能な単調増加な関数 $f : \mathbb{N} \to \mathbb{N}$ が存在して，

$$P((\exists n \geq f(m))[|S_n/n - p| > 2^{-m}]) < 2^{-m}.$$

積率母関数を使ってもう少し頑張ると，次のことが示せる．

補題 7.4 (Chernoff 上界の簡単な形)**.**

$$P(|S_n/n - p| > \epsilon) < 2\exp\left(-\frac{n\epsilon^2}{3p}\right)$$

よって，

$$P((\exists n \geq k)|S_n/n - p| > 2^{-m}) \leq \sum_{n=k}^{\infty} P(|S_n/n - p| > 2^{-m}) = \frac{2\exp(-k2^{-2m}/3p)}{1 - \exp(-2^{-2m}/3p)}$$

k を十分大きく取れば，最後の値は 2^{-m} よりも小さくできる．例えば，$k = Cm2^{2m}$ くらい？

7.2　二項分布の正規分布による近似の証明

二項分布が正規分布で近似されることを示そう．二項分布 $B(n, p)$ において，$q = 1 - p$ として $Y = \frac{X-np}{\sqrt{npq}}$ を考える．$X = k$ のとき $Y = t$ となるとすれば，当然 $P(Y = t) = P(X = k)$ となる．Y の確率分布は $\sqrt{npq}$ ごとに分かれていることを考えると，極限の確率密度関数としての値は，

$$f_n(t) = \sqrt{npq}\frac{n!}{k!(n-k)!}p^k q^{n-k}$$

$n \to \infty$ としたときのこの値が，$\frac{1}{\sqrt{2\pi}}\exp(-t^2/2)$ であることを示す．

まずスターリングの公式 $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ より，

$$f_n(t) \approx \sqrt{npq}\frac{\sqrt{2\pi n}}{\sqrt{2\pi k}\sqrt{2\pi(n-k)}}\left(\frac{np}{k}\right)^k\left(\frac{nq}{n-k}\right)^{n-k}$$

となる．ここで $t = \frac{k-np}{\sqrt{npq}}$ より，$\frac{k}{n} \to p$, $\frac{n-k}{n} \to q$ だから，前半部分は $\frac{1}{\sqrt{2\pi}}$ に収束する．そこで，後半部分を

$$g_n(t) = \left(\frac{np}{k}\right)^k\left(\frac{nq}{n-k}\right)^{n-k}$$

とおくと，

$$\frac{k}{np} = 1 + t\sqrt{\frac{q}{np}}$$

より，

$$\log\frac{k}{np} \approx t\sqrt{\frac{q}{np}} - \frac{t^2 q}{2np}$$

同様にして，

$$\frac{n-k}{nq} = 1 - t\sqrt{\frac{p}{nq}}$$

より，

$$\log\frac{n-k}{nq} \approx -t\sqrt{\frac{p}{nq}} - \frac{t^2 p}{2nq}$$

よって，

$$\begin{aligned}\log g_n(t) \approx & -(np + t\sqrt{npq})(t\sqrt{\frac{q}{np}} - \frac{t^2 q}{2np}) - (nq - t\sqrt{npq})(-t\sqrt{\frac{p}{nq}} - \frac{t^2 p}{2nq}) \\ = & -t\sqrt{npq} + t\sqrt{npq} - t^2 q - t^2 p + \frac{t^2 q}{2} + \frac{t^2 q}{2} + \frac{t^3 q^{3/2}}{2} - \frac{t^3 p^{3/2}}{2} \\ \approx & -\frac{t^2}{2}\end{aligned}$$

よって，$f_n(t) \to \frac{1}{\sqrt{2\pi}} \exp(-\frac{t^2}{2})$.

7.3 χ^2 分布

定理 7.5. $X_1, X_2, \cdots, X_n$ が独立に標準正規分布に従うとき，$X = X_1^2 + X_2^2 + \cdots + X_n^2$ は自由度 n のカイ二乗分布に従う．

定義 7.6. 自由度 n のカイ二乗分布は，密度関数が，

$$f_n(x) = \frac{1}{2^{n/2}\Gamma(n/2)} x^{n/2-1} e^{-x/2}$$

で表される関数である．

$\Gamma(n/2)$ は規格化定数のために必要なものと理解すれば良い．

定理 7.7. 独立な確率変数 X, Y の密度関数を f, g とする．$X + Y$ の密度関数は，$\int_{-\infty}^{\infty} f(t)g(x-t)dt$ で表される．

証明. (X, Y) の同時密度関数は $f(s)g(t)$ だから，$Z = X + Y$ の累積分布関数は，

$$D = \{(s, t) \ : \ s + t \leq x\}$$

として，

$$F_Z(x) = \int\int_D f(s)g(t)dsdt = \int_{-\infty}^{\infty}\int_{-\infty}^{x} f(s)g(u-s)dsdu = \int_{-\infty}^{\infty} f(s)F_Y(x-s)ds$$

より，

$$f_Z(x) = \int_{-\infty}^{\infty} f(s)g(x-s)ds$$

□

定理 7.5 の証明. まず，$n = 1$ の場合を示す．X は $N(0, 1)$ に従うとして，$Y = X^2$ とする．Y の累積分布関数は，$x \geq 0$ として，

$$F_Y(x) = P(Y \leq x) = P(X^2 \leq x) = \int_{-\sqrt{x}}^{\sqrt{x}} \frac{1}{\sqrt{2\pi}} \exp(-t^2/2)dt$$

よって，密度関数は，

$$f_Y(x) = 2\frac{1}{2\pi}\exp(-x/2)\frac{1}{2\sqrt{x}} = \frac{1}{2^{1/2}\sqrt{\pi}} x^{-1/2}\exp(-x/2).$$

$\Gamma(1/2) = \sqrt{\pi}$ で，これは自由度 1 のカイ二乗分布．

次に一般の場合は，帰納法で示す．Y が自由度 $n-1$ のカイ二乗分布に従い，Z が自由度 1 のカイ二乗分布に従うとする．$X = Y + Z$ の密度関数は，

$$\begin{aligned}\int_0^x f_{n-1}(t)f_1(x-t)dt = & \int_0^x \frac{1}{2^{(n-1)/2}\Gamma((n-1)/2)} t^{(n-1)/2-1} e^{-t/2} \frac{1}{2^{1/2}\sqrt{\pi}} t^{-1/2} e^{-(x-t)/2} dt \\ = & \frac{e^{-x/2}}{2^{n/2}\Gamma((n-1)/2)\sqrt{\pi}} \int_0^x t^{(n-3)/2}(x-t)^{-1/2} dt \\ = & \frac{\int_0^1 u^{(n-3)/2}(1-u)^{-1/2} du}{2^{n/2}\Gamma((n-1)/2)\sqrt{\pi}} x^{n/2-1} e^{-x/2}\end{aligned}$$

と書ける．これは自由度 n のカイ二乗分布である． □

7.4　適合度検定

確率変数 X が取る値を $E_1, E_2, \cdots, E_k$ とし，これらの取る値を $p_1, p_2, \cdots, p_k$ とする．n 回の試行の後，$E_1, \cdots, E_k$ に表れる回数を $n_1, n_2, \cdots, n_k$ とすると，

$$\chi^2 = \sum_{i=1}^{k} \frac{(n_i - np_i)^2}{np_i}$$

の従う分布は，n が十分大きい時，自由度 $k-1$ の χ^2 分布で近似できる．

ここでは，$k = 2$ の場合だけ証明を与える．一般的な k についての証明は，共分散の計算など，少し面倒になる．

E, F を取る確率が p, q であり，それぞれの回数を n_e, n_f とすると，

$$\begin{aligned}\chi^2 =& \frac{(n_e - np)^2}{np} + \frac{(n_f - nq)^2}{nq} \\ =& \frac{q(n_e - np)^2 + p(n_f - nq)^2}{npq}\end{aligned}$$

この分子の部分は，

$$(\text{分子}) = (1-p)(n_e - np)^2 + p(n - n_e - n(1-p))^2 = (n_e - np)^2$$

なので，

$$\chi^2 = \left(\frac{n_e - np}{\sqrt{npq}}\right)^2$$

中心極限定理より，これは標準正規分布で近似でき，それは自由度 1 の χ^2 分布である．

7.5　Martin-Löf 検定

定義 7.8 (Martin-Löf 1966)**.** 一様 c.e. 集合 (計算可能に近似可能) の列 $\{U_m\}$ で $\mu(U_m) \leq 2^{-m}$ となるものを，*Martin-Löf* **検定** (ML 検定) と呼ぶ．すべての ML 検定に合格する列を ML ランダムな列と呼ぶ．

定理 7.9. *ML* ランダムな列は絶対正規数である．特に大数の法則を満たす．

定理 7.10. 計算可能な列は *ML* ランダムではない．

8　第 8 回　乱数の定義：有限列の情報量

8.1　実数の名前

すべての自然数には名前があり，1, 2, 3 などと呼ばれる．有理数にも名前があり，1/2, 4/5 などと表現される．実数の中にも名前のあるものはたくさん存在する．例えば，$\sqrt{2}, \pi, e$ など．これらはそれぞれ，1 つの実数を表現しており，その意味で完全な規則を表しているものである．

数とその名前は 1 対 1 には対応しない．例えば，

$$1,\ 0.9999\cdots$$

という 2 つの表現は全く同じ数を指している．同じ数に 2 つの名前が存在するということである．$0.9999\cdots$ はいつまでたっても 1 に等しくはならないのではと思う人があるかもしれない．$0.9999\cdots$ という表現は，その近づく先の数を表しているのであり，それは 1 に完全に等しい．

すべての実数は無限の長さの表現で表すことができるが，以下では有限の表現という意味で名前という言葉を使う．名前を持たない実数が存在することを示そう．なぜ名前を持ちえないのか．名前や規則は可算個しか存在し得ないのに対し，実数は非可算個存在し，実数が多すぎるからである．そのことをもう少し詳しく話そう．

日本語では数は，一，十，百，千，万，億，兆，京，垓などと表現される．塵劫記 (1627) では無量大数までが紹介されている．仏典の華厳経には更に大きな数まで記述されている．大きな数に名前をつければ，その数まで (原理的には) 数えることができるようになる．言語によっては 10 程度までの数しかない言語もあり，その言語では大きな数は数えられない．アラビア数字による位取りの記法を用いれば，どこまででも大きな数を数えられるようになり，また便利なため，数学の発展に大きく寄与している．

今，2 つの集合がどちらが多いかを考える．有限の場合には，それぞれの数を数えて，比較すれば良い．では無限の場合にはどのようにしたら良いか？

定義 8.1. 集合 A と B の**基数が等しい**とは，A, B の間に全単射が存在することを言う．

英語で one, two, three と first, second third の違いがあるが，前者は基数詞と呼ばれ量を表すのに対し，後者は順序数詞と呼ばれ順序を表す．全単射とは，異なるものは異なるものに対応づけられ，余るものがない状態を指す．

例えば，$A = \{1, 2, 3\}$ と $B = \{4, 5, 6\}$ は基数が等しい．1 対 1 の対応が存在するからである．

定理 8.2. $\mathbb{N}$ と $\mathbb{Z}$ は基数が等しい．

証明.

$$\mathbb{Z} = \{0, 1, -1, 2, -2, 3, -3, \cdots\}$$

□

定理 8.3. $\mathbb{N}$ と $\mathbb{Q}$ は基数が等しい．

証明. $\mathbb{Q}$ と格子点を同一視して，順に並べれば良い． □

定理 8.4. $\mathbb{N}$ と $\mathbb{R}$ は基数が等しくない．

証明. $(0, 1)$ を並べることができたとして，それを x_n とおく．x_n を 2 進展開して，m 桁目を $x_{n,m}$ とおく．y_n として，1 から 8 までの整数の中で，$x_{n,n}$ と異なるものとする．すると，$y = \{y_n\}$ は 1 つの実数の 2 進展開を表すが，それはどの x_n とも n 桁目を見れば異なる．

一方，$(0, 1)$ と $(-1, 1)$ は，$y = 2x - 1$ という関数で全単射が存在する．さらに，$(-1, 1)$ と $\mathbb{R}$ は，$y = \tan \frac{\pi x}{2}$ という関数で全単射が存在する． □

ここで分かったことは無限にも種類があるということである．その中で一番小さなものが自然数の個数で**可算濃度**と呼ばれる．実数の個数を**連続濃度**と呼ぶ．

8.2　名前の複雑さ

前節の結果から，ほとんどの実数は名前 (有限の表現) を持たないことが分かる．では最初の n 桁を表現するのにはどの程度の長さの表現が必要なのだろうか？以下のように定義を行うと，万能機械の存在から，圧縮可能性の概念を導くことができる．

U を $2^{<\omega}$ から $2^{<\omega}$ の計算可能関数とする．さらに，どんな計算可能関数をもシミュレートできる万能関数であると仮定する．さらに，その値域が prefix-free であると仮定しよう．prefix-free とは，電話番号のように，どこまででプログラムが終了するかが判定できるプログラミング言語だと思ってもらって良い．

定義 8.5.

$$K(\sigma) = K_U(\sigma) = \{|\tau| \ : \ U(\tau) = \sigma\}.$$

σ を出力する入力のうちで最小の長さを σ の複雑性と呼ぶ．σ が元のファイル，τ が圧縮されたファイル，M が解凍アルゴリズムと思えば良い．

例えば，00000000111111111111111110000000000000000111111111 のように 0 や 1 が長く続く文字列の場合には，8,10,13,10 のようにその長さを数字で表し，それを符号化したほうが短くなる．n 桁の長さを 2 進数で表せば，$\log_2 n$ 桁になり，区切りのために冗長化しても，$2\log_2 n$ 桁なので，n よりは短い．これを復号化するアルゴリズムを M とすれば，$C_M(\sigma)$ は σ を圧縮した長さとなる．0 や 1 が長く続く場合にはこのような圧縮方法があるが，長く続かない場合には，この方法で圧縮すると短くならない．規則があれば，それに合わせた圧縮プログラムがあって，短くできる．最近ではほとんどが zip 圧縮だが，圧縮アルゴリズムには多くのものが存在する．それはテキスト，音楽，画像，動画など，ファイルの特性に応じて圧縮するからである．

定理 8.6. どんなファイルも短く圧縮できる圧縮プログラムは存在しない．

証明. そのようなプログラム M が存在したとしよう．すると任意の $\sigma \in 2^{<\omega}$ に対して，σ を出力する入力 $\tau \in 2^{<\omega}$ で，σ よりも短いものが存在する．そのようなもののうちの 1 つを出力として返す関数を f とする．すると，

$$|\sigma| > |f(\sigma)| > |f^2(\sigma)| > \cdots > |f^{n-1}(\sigma)| > \cdots$$

となるがこのようなことは起こりえない． □

大雑把に言えば，圧縮されたファイルは圧縮しにくいということである．

別証明を見ておこう

証明. そのようなプログラム M が存在したとする．n 桁の文字列は 2^n 個あるが，それらがすべて短く圧縮できるとすれば，$n-1$ 桁以下の文字列で，n 桁を出力するプログラムが少なくとも 2^n 個必要である．しかし，$n-1$ 桁以下の文字列の個数は，

$$1 + 2 + \cdots + 2^{n-1} = 2^n - 1$$

なので，これは不可能である． □

このことから，ほとんどの文字列があまり圧縮できないことが分かる．特に，c 桁圧縮できる文字列は，全体のうち 2^{-c} くらいしか存在しない．

定理 8.7. $X \in 2^\omega$ が ML ランダムであることと，

$$K(X \upharpoonright n) > n - O(1)$$

であることは同値.

すなわちランダムな列とは圧縮できない列にほかならない．

8.3 単純 Kolmogorov 複雑性

さて，では「どんなプログラムでもほとんど圧縮できない文字列」は存在するのだろうか？それとも「どんな文字列も，適当なプログラムを見つければ圧縮できる」のだろうか？実はどんな圧縮プログラムにも劣らない圧縮プログラムが存在する．そのプログラムで圧縮できなければ，どんなプログラムでも圧縮できない．

定理 8.8. U を万能機械とすると，任意の機械 M に対して，

$$C_U(\sigma) \leq C_M(\sigma) + O(1).$$

すなわち U による圧縮は M による圧縮と比較して定数くらいしか悪くならない．

証明. U は万能機械なので，ある文字列 τ が存在して，$U(\tau\rho) = M(\rho)$．任意の σ に対して，σ^* を $M(\sigma^*) = \sigma$, $|\sigma^*| = K_M(\sigma)$ とすると，$U(\tau\sigma^*) = \sigma$ より，

$$C_U(\sigma) \leq |\tau\sigma^*| = C_M(\sigma) + |\tau|.$$

□

では，この圧縮プログラムは実際に使われているのだろうか．実は実用的ではない．

定理 8.9. $\sigma \mapsto C_U(\sigma)$ は計算可能ではない．

証明. $n \in \omega$ に対し，$x_n \in 2^{<\omega}$ を $C_U(x_n) \geq n$ を満たす辞書式で最小のものとする．$\sigma \mapsto C_U(\sigma)$ は計算可能なので，$n \mapsto x_n$ は計算可能．よって，$C_U(x_n) \leq \log n + O(1)$．これは十分大きい n に対しては成立しない． □

構造としては「19 文字以内で記述できない最小の自然数」という Berry の逆説と同じ形をしている．特に，ランダムな列がランダムであるかどうかは判定できない．

実用性はなくても，圧縮不可能性によりランダム性を捉えることで，ランダムな列の性質が分かるようになるという意味がある．

乱数列を「すべての検定に合格する列」として定義したい．前回の万能機械による圧縮検定は，そのような定義ができそうであることを示唆する．例えば，

$$C_U(\sigma) \geq |\sigma| - c$$

を満たす σ を，c-ランダムと呼ぶ，などの定義が考えられる．ところが，この定義は万能機械 U に依存しており不自然である．

このような問題は無限列では起こらない．そこで，次のような定義が考えられる．

$$C_U(X \upharpoonright n) \geq n - O(1)$$

ところが，今度はこのような $X \in 2^\omega$ は存在しないことが分かっている．

そのため考えられたのが prefix-free の複雑性で K で表される．

8.4 接頭離 Kolmogorov 複雑性

接頭離機械とは，入力の終了が判定できる機械のことをいう．通常のコンピュータでは，ファイルの終了は EOF や CTRL-d などで終了判定を行っている．このような特別な文字列を使わずに終了判定を行う必要がある．接頭離機械では，σ を入力として何かを出力すれば，σ を接頭辞に持つ別の文字列は何も出力しない．このようにして，入力の終了を判定する．このような考え方は電話番号などでも応用されている．

前回の方法と同様にすれば，以下のことが分かる．

定理 8.10. 任意の接頭離機械を模倣する接頭離万能機械 U が存在する．

そこで，この U を使って次のように Kolmogorov 複雑性を修正する．

定義 8.11.

$$K(\sigma) = K_U(\sigma) = \{|\tau| \ : \ U(\tau) = \sigma\}.$$

定義 8.12. $X \in 2^\omega$ が *ML* **ランダム**であるとは,

$$K(X \upharpoonright n) > n - O(1)$$

であることをいう.

このランダム性の定義は適切な概念 (の 1 つ) として広く認識されており，様々な良い性質を持つ．たとえば，2 進無限列のほとんど (一様測度の意味でほとんど至る所) が ML ランダムである．また，ML ランダムであれば大数の法則が成り立つ.

実は 2^ω と $(0,1)$ は基数が等しいことも分かる．また，実数のほとんどには有限情報での名前をつけることはできない.

ほとんどの実数の名前には可算無限桁の情報が必要である．ML ランダムであるとは，その桁数の発散速度がかなり早いことを意味している.

定理 8.13. 計算可能な実数は *ML* ランダムではない．特に，e, π は *ML* ランダムではない.

証明. $A \in 2^\omega$ が計算可能であれば,

$$K(A \upharpoonright n) \leq K(n) + O(1) \leq 2 \log n + O(1).$$

□

8.5 停止問題

今日は乱数列の例として停止確率 Ω を紹介する．しかし，その前に停止問題 H の計算不可能性を見ておく.

定理 8.14. 停止問題 $H = \{(e,k) \ : \ \Phi_e(k) \downarrow\}$ は計算可能ではない.

証明. 集合 $K = \{e \ : \ \Phi_e(e) \downarrow\}$ は計算可能ではないことを示せば十分．K が計算可能であれば,

$$f(n) = \begin{cases} \Phi_n(n) + 1 & (n \in K) \\ 0 & (n \notin K) \end{cases}$$

も計算可能となる．f の指数を e' とすれば，$e' \in K$ ならば,

$$\Phi_{e'}(e') = f(e') = \Phi_{e'}(e') + 1$$

となり，$e' \notin K$ ならば,

$$\Phi_{e'}(e') = f(e') = 0$$

となり，これは $e' \notin K$ に反する. □

停止問題が計算出来ないことの身近な例として，停止しないプログラムを予めは決定できないことなどが挙げられる.

停止問題は計算出来ない問題であるから，停止問題が与えられたとすれば計算できるような問題は，ない場合に比べて広くなる.

8.6 停止確率 Ω

定義 8.15. 最適機械 U に対して，**停止確率** *(halting probability)*Ω_U を

$$\Omega_U = \sum_{\sigma \in \mathrm{dom}(U)} 2^{-|\sigma|}$$

により定義する．最適機械 U に依存しない性質について語る場合には，U を省略して，単に Ω と書く.

定理 8.16. $\Omega \equiv_T \emptyset'$.

証明. Ω は左計算可算数なので $\emptyset'$ で計算できる.

次に $\emptyset'$ を Ω を使って計算できることを示そう. 機械 M を $M(0^e1)\downarrow = \lambda \iff \Phi_e(e)\downarrow$ として定義する. U は最適なので, ある $\rho \in 2^{<\omega}$ が存在して, $U(\rho 0^e 1)\downarrow \iff e \in \emptyset'$ が成り立つ. Ω を使って $e \in \emptyset'$ かどうかを判定するには, $\Omega - \Omega_s < 2^{-(|\rho|+e+1)}$ となる段階まで待つ. もし $U(\rho 0^e 1)[s]\downarrow$ であれば, $M(0^e1)\downarrow$ であり, $e \in \emptyset'$ である. $U(\rho 0^e 1)[s]\uparrow$ とする. もし段階 $t > s$ で $U(\rho 0^e 1)[t]\downarrow$ となれば,

$$\Omega_t \geq \Omega_s + 2^{-(|\rho|+e+1)} > \Omega$$

となり矛盾するので, $U(\rho 0^e 1)\uparrow$ である. すなわち $e \notin \emptyset'$. □

定理 8.17. Ω は 1 ランダムである.

証明. Ω は有理数ではないので, すべての n に対して段階 s が存在して $\Omega_s \upharpoonright n = \Omega \upharpoonright n$ となることに注意しておく.

機械 M を定義する. 再帰定理より M の指数を使って良く, U では $U(0^c1\sigma) = M(\sigma)$ のように表現できるとする. 入力 $\tau \in 2^{<\omega}$ に対して, $U(\tau)[s] = \Omega_s \upharpoonright n$ かつ $|\tau| < n - c$ (これは $K(\Omega_s \upharpoonright n)[s] < n - c$ を意味している) であったならば, $\mu \notin \mathrm{rng}U[s]$ となる $\mu \in 2^{<\omega}$ を見つけ, $M(\tau) = \mu$ とする.

このとき, $U(0^c1\tau) = \mu$ で $U(0^c1\tau)[s]\uparrow$ より, $\Omega - \Omega_s \geq 2^{-(c+1+|\tau|)} > 2^{-n}$ であり, $\Omega \upharpoonright n \neq \Omega_s \upharpoonright n$. すなわち, $|\tau| < n - c$ ならば $U(\tau) \neq \Omega \upharpoonright n$ であるので, $K(\Omega \upharpoonright n) \geq n - c$. □

役に立つ ML ランダム列が存在する！！

8.7 独立性定理

定理 8.18. *$X \oplus Y$ が ML ランダムであることと, X が ML ランダムかつ Y が X-ML ランダムであることは同値.*

証明. Y が X-ML ランダムでなければ, 任意の d について,

$$K^X(Y \upharpoonright n) < n - d$$

となる n が存在する. ここで,

$$\{\langle \sigma, \tau \rangle \ : \ K^\sigma(\tau) < |\tau| - d, \ |\sigma| \geq |\tau|\}$$

を考え, それぞれの τ について σ は接頭離集合となるように部分集合を考える. それぞれのプログラム ρ に対して, ρ,$\overline{d}$,σ,x の結合を入力とし, $\sigma \oplus (\tau x)$ を出力とするような機械を考える. ここで, $\overline{d}$ は $d \in \omega$ を表現する長さ $2\log_2 d + 2$ の文字列であり, x は $|x| = |\sigma| - |\tau|$ となるような任意の文字列である. この機械は接頭離である. なぜなら, ρ から τ が, $\overline{d}$ から d が計算できるので, σ はその文字列を見た時に入ることが分かるからである. すると, $X \oplus Y \upharpoonright (2|\sigma|)$ の文字列が長さ $d - (2\log_2 d + 2)$ の圧縮ができることが分かる.

X が ML ランダムかつ Y が X-ML ランダムであるとする.

$$K(X \oplus Y \upharpoonright 2n) > 2n - O(1)$$

を示せば良い. n を十分大きな数として固定する.

$$K(X \upharpoonright n) = n + c_n$$

とおくと,

$$l_n = K(Y \upharpoonright n | (X \upharpoonright n)^*) > n - c_n - O(1)$$

を示せば良い. それは,

$$n - O(1) \leq K^X(Y \upharpoonright n) \leq l_n + K(|l_n - n|) + K(c_n)$$

より得られる. □

9　第 9 回　乱数の応用：暗号

9.1　講義概観

講義も半分が終わり折り返し地点となった．この講義がどこに向かっているのか分からないという質問があった．最初にゴールを明らかにするのは大事なことであり，第 1 回の時点でも少しは話をしたが，半分が終わった段階でもう少し詳しく話ができるようになった．そこで，これまでの講義を振り返りつつ，これから話すことを概観しよう．

まず，第 1 回では「乱数は様々な場面で使われる」ことを話した．具体的には，ゲーム，ランダムに選ぶ，確率モデルのシミュレーション，乱択アルゴリズム，暗号などに使われるという話をした．乱数は現代社会の基盤の 1 つである．

第 2,3 回では「乱数の作り方」を話した．一様乱数として，線形合同法，メルセンヌ・ツイスタ，Xorshift を話した．一般の乱数として，逆関数法，極座標法について話した．実際に使われている乱数が意外に単純な方法で実装されていることに驚いたかもしれない．

第 4,5 回では「乱数の検定の仕方」を話した．乱数が持つ統計的性質に着目し，統計的仮説検定の考え方を紹介した．乱数とは「ある確率分布の標本点として不自然でないものである」という考え方について理解を深めて欲しかった．

第 6,7,8 回では「乱数とは何か」という問題に数学的定義を与えて，その性質を見ることが目標である．第 1 回から第 5 回までは工学的な視点で「どうすればよいのか」という具体的な手法の紹介に焦点が置かれていたのに対し，第 6 回から第 8 回では「そもそも乱数とはどういうもので，どういう性質を持つのか」という理学的な視点で話をしている．そこで得られた知見を元に，第 9 回以降では「乱数であることを保証する」という話をする．理学的な基盤を元にもう一度工学的な手法を改良したり，限界を議論したり，その性質を保証したりする．このような視点は，暗号や署名などの応用で重要になる．この講義は理工学研究科総合講義なので，細かい議論には目をつぶって，理学的な視点と工学的な視点がどのように融合されるかという話がしたいと思って，このテーマを選んでいる．

さて，乱数とは「圧縮不可能な列」として定義できる．圧縮という検定は万能検定と見なせる．このことは，万能機械の存在から出てくる．乱数を定義するためには，計算可能という概念を考える必要があるのである．すなわち，計算によって規則が見つけられない列が乱数である．この計算可能と存在という概念のギャップが乱数の本質である．

「完璧な乱数は存在するか？」という質問をよく受ける．完璧な乱数というものを心のなかに思い浮かべている人も多いようである．完璧な正三角形がイデアに存在すると仮定するように．神は完璧な乱数を作れるか，などと質問する人もいる．

これまでの乱数の研究が教えるところは，乱数は概念のギャップに存在するということである．だから上のような質問は意味が無い．このギャップを理解するために，可算と非可算の概念を紹介した．そこで使われた Cantor の対角線論法，すなわちそのギャップの間にある存在の構成は，その後の多くの理論の基礎となっている．今日第 8 回では更に乱数の性質を見る．

第 9 回以降では「擬似乱数」について話をする．数学的な乱数の定義は可能であったが，その性質上，計算機で構成することはできない．応用上は作れる乱数に興味がある．乱数の本質はギャップにあった．そこで「現実的な計算時間では真の乱数と見分けがつかない数列」を考える．そのような列が擬似乱数である．

このような定義を行うためには，計算時間によって問題の困難さを分類する計算量理論についての知識が必要である．第 9 回では計算量理論の導入を行う．第 10 回では乱数を使うと計算量的な困難さが変わる問題を考える．いわゆる乱択アルゴリズムである．第 11 回では一方向関数を導入して，計算量におけるギャップについて説明する．第 12 回以降では擬似乱数生成器や暗号など，計算量理論に基づいた乱数の応用について見る．

9.2　乱数の暗号への応用

Alice から Bob へ情報を伝えたい．その通信路は盗聴が可能である．他の人に知られないようにするために，情報の暗号化が必要である．

今，私たちは ML ランダムという完璧な乱数の存在をしっている．この乱数を使って，次のようにして完璧な暗号

化が可能である．

(1) Alice が送信したい文章を 2 進有限列 A に変換する
(2) 鍵 (password) としてそれと同じ長さのランダムな列 X を用意して，xor をとり $A \oplus X$ として暗号化する
(3) 暗号化した文字列 $A \oplus X$ を Bob に送る
(4) 何らかの方法で Bob に鍵を送る
(5) Bob は $(A \oplus X) \oplus X = A$ として復号化し，さらに通常の文章に変換する．

この方法は数学的に完全に安全であることが保証できる．すなわち鍵が知られない限り，$A \oplus X$ から A を推測することはできない．推測できるのは文章の長さだけである．それも推測されるのが嫌なのであれば，適当な文字列を付け加えて長くすれば良い．推測できないことを保証する性質こそ，X のランダム性である．この意味でランダム性はとても重要な性質である．

ところが，実際には上の手法は使えない．大きな問題が 2 つある．

1 つは鍵配送問題と呼ばれる．どうやって鍵を送るのかという問題である．隣の席に座っているのであれば，USB で渡せば良いかもしれない．しかし暗号が使われる現在の典型的な例は，インターネット上でクレジットカードの番号を他の人に知られないように送る，という類のものである．インターネットでの通信が安全でないから暗号化したいのである．その鍵をネットで送るのは意味がない．かといって，毎回郵送するのであれば，なんのためのインターネットか分からない．これを解決したのが公開鍵暗号であり，今日のテーマである．

もう 1 つは乱数をいかにして作るかという問題である．鍵は少なくとも文章以上の長さが必要になる．毎回同じ鍵を使うと，そのことにより鍵そのものが予想できるので，毎回異なる鍵を使う必要がある．そうすると，鍵の長さが膨大になる．携帯のログインパスワードが 1000 文字とかであれば，実用的でないのと同じである．やはり鍵は何らかの方法で計算可能でなければならない．その場合には，暗号を破ることもまた計算可能になり，安全性が担保できなくなる．そこで必要になってくるのが，文章の暗号化は簡単に計算できるが，暗号を破る計算は難しい，という計算の難易度の非対称性である．素因数分解の困難性を使った暗号である RSA 暗号を解説し，その後更なる保証のために必要な数学的枠組みを解説しよう．

9.3 公開鍵暗号 RSA

暗号化と復号化に同じ鍵を使う方式を秘密鍵暗号方式と言う．通常のコンピュータや携帯，銀行の ATM などで使うパスワードでは，設定した鍵と同じ鍵でログインできる．普通は短い鍵から一意的に生成される乱数により暗号化するなどして使う．インターネット上のサービスにログインする場合，適当なパスワードを設定することが多い．そのパスワードはサービス提供側のサーバーに保存されている．それが何らかの方法で流出すれば問題が起こる．またそのサービス提供側の中に悪意を持ってその情報を使う人がいることも考えられる．そのため，普通は Hash 関数の値だけを保存して，それが一致するかどうかによって，ログインを許すかどうかを判定することが多い．

しかし，この場合は鍵配送問題が起こる．これを解決したのが公開鍵暗号方式で，セキュリティの分野では最も大きな発明と言って良いだろう．

公開鍵暗号方式では，2 つの鍵が使われる．公開鍵とプライベート鍵の 2 つで，公開鍵は一般に公開するもの，プライベート鍵は秘密にしておくものである．公開鍵は暗号化のために使われ，プライベート鍵は復号化のために使われる．

今，Alice から Bob に秘密のメールを送りたいとする．Bob は何らかの方法で，公開鍵とプライベート鍵を作る．公開鍵を公開する．これは「私 Bob に秘密のメールを送りたいときは，この公開鍵を使ってね」という意味である．Alice はこれを見て，その公開鍵を使って暗号化し，その文字列を Bob に送る．Bob はプライベート鍵で復号化して読む．暗号化された文字列は，プライベート鍵を持たない Alice も読むことができない．

ここで必要となるのは次の性質である．

(1) 公開鍵で暗号化された文字列は，プライベート鍵で復号化できる．
(2) 公開鍵からプライベート鍵は推測が難しい．

公開鍵とプライベート鍵は深い関係があるはずである．それにも関わらず，推測が難しいという関係を保つ必要がある．そんなことが可能なのだろうか．具体例として，現在使われている RSA 暗号を例に見てみよう．

1977 年に発明されて，Ron Rivest, Adi Shamir, Len Adleman の頭文字で RSA と呼ばれている．彼らは 2002 年にチューリング賞を受賞している．ただし，裏の歴史もある．それ以前にイギリスの最高機密期間 GCHQ の職員のエリスとコックスにより発見されているが，機密事項とされたため世に知られることはなかった．

送りたい文章を適当な変換により数字にする．p, q として大きな素数をとる．$n = pq$, $l = (p-1)(q-1)$ e は l と互いに素で l より小さい．$ed \equiv 1 \mod l$ となるように d を選ぶ．公開鍵は (e, n)，プライベート鍵は (d, n) である．暗号化は文章を k として，

$$c = k^e \mod n$$

とする．復号化は，

$$c^d \mod n$$

とすると，k になる．

なぜ復号化で k になるのか？フェルマーの小定理の拡張であるオイラーの定理から，

$$c^d \equiv k^{ed} \equiv k^{l+1} \equiv k \mod n$$

となるから．

素数であることはどうやって判定するの？ランダムに数を選んで，Miller-Rabin 素数判定法で判定する．

ランダムに選ぶ必要はあるの？ランダムに選ばなければ，p, q がわかり，d も計算できてしまいますよ！

e や d はどうやって選ぶの？そもそもそういう数は存在するの？ p, q が素数であれば必ず存在する．p, q がわかっていれば，拡張ユークリッドの互除法で素早く計算できる．

(e, n) が分かれば，p, q もわかって，l も計算できて，e も計算できるから，暗号になっていないのでは？素因数分解はとても難しいことが知られていて，実用的な時間での計算は不可能．

これまでは n として 512 ビットの暗号化が使われていた時代もあるが，すでに使用推奨期間が切れている．1024 ビットの使用推奨期間は 2010 年，2048 ビットは 2030 年．それ以降も安全に使いたいのであれば，少なくとも 3072 ビットの RSA を使うようにと言われている．

RSA 暗号は素因数分解の困難性に依拠している．この困難性が克服されれば，RSA 暗号は破られてしまう．今のところ安全なだけで，安全の保証がされているわけではない．

数学的に安全であることを保証できるような暗号化は作れるだろうか？そのような暗号化は何が作れたらできるのか，どんな性質を持つべきなのか，何をもって保証できるのか．来週以降はその検証のために必要な数学を紹介する．

擬似乱数の応用として，暗号や署名が存在することを説明する．

9.4 秘密鍵暗号

送りたいメッセージをコード化したものを平文という．これを盗聴が可能な通信経路を使って，遠く離れた人に送りたいとしよう．ハガキに文字を書いて送るときには，誰の目に触れるか分からない．インターネットでメールを送るのはハガキに書いて送るようなもので，中継地の人は全部読めてしまう．

そこで，平文を暗号化して送る．ちょうど封書に入れるようなものである．これを受け取った人は何が書かれているか分からない．しかし受け取った人はこの暗号を平文に復号できて欲しい．そのためには復号化のための鍵が必要である．

この暗号方式として様々なものが考えられてきた．最も有名なものはシーザー暗号と呼ばれるもので，アルファベットで書かれた平文を何文字かずらして送り，復号化する場合には逆にずらすというものである．この方法は一応は暗号っぽく見えるが，解読するための様々な方法が知られている．シーザー暗号の場合は，高々 26 種類試せば良いのだから，コンピュータがあればすぐに解読できる．何らかの方法でコード化して，種類を増やすことも考えられるが，英語でのアルファベットの出現頻度から，ずらす量は簡単に推測できる．このような単純な方法では，暗号としては役に立たない．

この講義では，「実際にどのような暗号方式が使われているか」ではなく，「理想的な暗号方式はどのような性質を持つのか」を考えよう．暗号の方式が公開されていたとしても，鍵が分からなければ，暗号文から平文が分からない，という状況が必要である．平文の一部さえも分かっては困る．しかし，鍵が分かれば暗号化も復号化も素早くできる必要がある．この状況は前回までやってきた状況によく似ていることが分かるだろう．この状況を前回のように厳密に定義できるが，ここでは省略することにしよう．

実際，擬似乱数関数から安全な秘密鍵暗号を作れる．$x \in 2^n$ を平文として，種 s を秘密鍵とする．$r \in 2^n$ を適当に選んで，$(r, x \oplus f_s(r))$ を暗号文とする．復号は (r, y) に対して，$y \oplus f_s(r)$ を計算すれば良い．s が分からなければ，r と y からは x は分からないことに注意しよう．

9.5 公開鍵暗号

秘密鍵暗号は鍵の受け渡しさえクリアすれば安全な暗号となるが，どうやって鍵を受け渡しするかが大きな問題となる．そこで考えられたのが公開鍵暗号である．

公開鍵暗号は 3 つのステップ (G, E, D) からなる．G は 1^n を入力として，(pk, sk) という 2 つの鍵の生成を行う．pk は public key で公開鍵と呼ばれる．sk は secret key で秘密鍵と呼ばれる．公開鍵は全体に公開するもので，秘密鍵は自分しか知らないようにしておく．この G を行うのは，平文を受け取る側である．平文を送る側は，この公開鍵 pk を見て，暗号化 (encryption) を行う．この暗号文を受け取った側は秘密鍵 sk を見て，復号化を行う．

それぞれの計算は多項式時間計算可能である必要がある．更に公開鍵と暗号文を見ても，秘密鍵と平文共に計算出来ない，という状況が必要である．これもまた一方向関数の応用として作ることができる．現実には RSA など素因数分解の困難さを利用する暗号が使われている．

9.6 電子署名

誰かに平文が送られた時に，それが途中で改竄されていないかを確認するという方法も，この応用で得ることができる．印鑑やサインでは，簡単に真似できるから，署名としての役割を果たさない．

公開鍵暗号の応用で，送る側が秘密鍵でハッシュを暗号化して平文と共に送る．受け取る側は公開鍵で復号化してハッシュが一致するか調べる．途中で改竄されていた場合は，ハッシュが異なり，検知できる．秘密鍵を知らなければ，新たな署名を作ることもできない．

10　第 11 回　擬似乱数：計算複雑性クラス

10.1　問題の困難さ

前回までで「計算によって規則を見つけられない列が乱数列である」という話をした．しかし，この定義では計算可能な列はランダムではないことになり，乱数列を計算によって作ることはできなくなってしまう．そこで，少し条件を緩めて，「素早い計算によって規則を見つけられない列」を**擬似乱数**と呼ぶことにしよう．そのような列を素早く計算できるだろうか？

この素早い計算とはどういうことかを理解するために，以下の問題を考えよう．ある問題に対して，

- 答えを見つける
- 答えが正しいかどうかを確認する

のは，同じ難しさであろうか．

数学の演習問題や試験などで，答えを見ずに解くよりは，答えが正しいことを確認することは容易であろう．半年か 1 年か授業で習うことを理解することは決して楽なことではないが，それを知らない人が試行錯誤して発見するよりは，ずっと早く高いレベルに到達できる．

10.2　SAT 問題

このことを数学的に表現してみよう．そのために次の**充足可能性問題** (satisfiability problem) を考えよう．通常，SAT 問題と言われる．そのために次のような論理式を考える．

$$\neg x_1 \wedge (x_2 \vee x_1)$$

$x_1, x_2, x_3, \cdots$ などの論理変数には TRUE か FALSE が入る．$\wedge$ は「かつ」の意味で，$\vee$ は「または」の意味，$\neg$ は否定の意味．これらを適当に組み合わせたものを命題論理式という．このような命題論理式が与えられた時に，論理変数に適当に T,F を入れて，論式の真偽値が T になるようにできるかどうか (充足可能かどうか) を判定する問題を SAT 問題という．上の問題であれば，$x_1 = F, x_2 = T$ とすれば良い．

SAT 問題は計算可能である．出てくる変数の数を n とすれば，論理変数の真偽値の組み合わせは 2^n なので，それらを全部計算すれば，充足可能かどうかは判定できる．

では，どれくらいの速さで計算できるだろうか．出てくる論理記号の数を k として，各論理式の計算に 1 ステップかかるとすると，1 つの論理変数の組の真偽を計算するのには k ステップかかる．論理変数の数の最大は $k+1$ なので，全部の計算を行うと，$k2^{k+1}$ ステップとなる．

もっと賢い方法はないのだろうか？例えば，別の方法で，$10 \cdot 2^k$ ステップでできる方法があったとする．どちらのほうが速いと言えるだろうか．$k = 1$ のときには，$k2^{k+1} < 10 \cdot 2^k$ であるが，このような比較は意味が無い．ステップと言っても，同じ時間かかるわけではないかもしれない．しかし，k を十分大きくとって，

$$\frac{k2^{k+1}}{10 \cdot 2^k} = \frac{k}{5} \to \infty$$

であることを考えると，$k2^{k+1}$ の方が早く発散する．そこで，このような意味で早く発散する方が時間がかかると考えよう．そうすれば，速いコンピュータと遅いコンピュータで考えた場合でも，アルゴリズムの性能のほうが重要になる．

一般に，

$$n \ll n^2 \ll n^3 \ll \cdots \ll 2^n \ll$$

となる．n が小さい時には大差なくても，n が大きくなると，この差はどんどん広がる．2^n ステップかかるアルゴリズムは，実際に計算することはほぼ不可能である．そのことを教えた動画が『組合せ爆発「フカシギの数え方」』なので一緒に見よう．このような背景から，多項式時間で計算可能＝実際に計算可能，計算には指数時間かかる＝実際には計算不可能，という考え方が一般的になった．

多項式時間で計算可能な問題を P と呼ぶ．また答えが与えられていれば，その確認が多項式時間で計算可能な問題を NP(Non-deterministic Polynomial time) と呼ぶ．SAT 問題は NP に入る．では NP に入るだろうか？これは未解決の問題である．もし SAT 問題が P に入らないことが分かれば，$P \neq NP$ が分かる．もし SAT 問題が P に入ることが分かれば，$P = NP$ が分かる．そのため，SAT 問題は NP 問題の中で最も難しい問題であり，NP 困難 (NP-hard) と呼ばれる．P と NP が真に異なるかという問題は，今後話すように乱数や暗号に関わる超重要問題であるが未解決である．2000 年にクレイ数学研究所のミレニアム懸賞問題の一つとして、この問題に対して 100 万ドルの懸賞金がかけられた。是非挑戦してもらいたい．

10.3　ハミルトン閉路問題と巡回セールスマン問題

このような NP 困難は様々な場面で自然にあらわれる．そのようなものの中で，ハミルトン閉路問題と巡回セールスマン問題を紹介する．

グラフ $G = (V, E)$ について，$s, t \in V$ に対して，s から t への**ハミルトン小路** (Hamilton path) とは，すべての頂点をちょうど 1 回ずつ訪れ，s で始まり t で終わる小路のことである．**ハミルトン閉路問題**は与えられたグラフに対して，ハミルトン小路となる閉路が存在するかどうかを判定する問題である．

ハミルトン閉路問題は NP 困難であることが分かっている．NP に属することは簡単に分かる．また，ハミルトン閉路が与えられた時に，対応する SAT の解を多項式時間で計算する手法を与えることで，NP 困難であることも証明できる．

巡回セールスマン問題は，グラフとその重み関数が存在して，重みが W 以下のハミルトン閉路が存在するかどうかを判定する問題である．この問題も同じく NP 困難であることが分かっている．

11　第 12 回　擬似乱数の定義：計算複雑性クラス

11.1　Miller-Rabin 素数判定法

n が素数かどうかを判定したい．素朴な方法では，2 から $\sqrt{n}$ の間の自然数で割って，どれかで割れれば合成数，いずれでも割れなければ素数，という判定法である．この判定には，どれくらいの時間がかかるだろうか．n は入力として，$k = \log n$ くらいなので，$\sqrt{n} = 2^{k/2}$ くらいの時間がかかる．通常の方法では指数時間のアルゴリズムということになる．

実は素数判定は多項式時間で計算できる，すなわち P であることが 2004 年に分かっている．しかし，実際に計算機で走らせるには非現実的な時間がかかる．

ここでは，Miller-Rabin 素数判定法を紹介しよう．まずは，フェルマーの小定理を思い出しておく．

定理 11.1. p が素数で，a と p が互いに素であれば，

$$a^{p-1} \equiv 1 \mod p$$

ところで，次のような事実が分かる．

命題 11.2.

$$a^2 \equiv 1 \mod p$$

ならば

$$a \equiv \pm 1 \mod p$$

なぜなら，$(a+1)(a-1) \equiv 0 \mod p$ となって，$a+1$ と p が互いに素であれば $a \equiv 1$ となり，$a+1$ と p が互いに素でなければ，$a+1$ が p の倍数で，$a \equiv -1$ となる．このことから次のことが分かる．

定理 11.3. p を素数とする．a は p と互いに素として，$p-1 = b2^d$, b は奇数, d は整数，とする．$s(a,i) = a^{b2^i}$ を考えると，

$$s(a,d) \equiv 1 \mod p$$

かつ

$$s(a,0) \equiv 1 \mod p \vee (\exists i \leq d-1) s(a,i) \equiv -1 \mod p$$

実は，p が素数でない時には，上の 2 条件を満たすような a は 1 から $p-1$ のうち，高々半分であることが分かっている．この事実を利用した以下の確率的素数判定アルゴリズムを Miller-Rabin 素数判定法という．

入力: 素数かどうかを判定する自然数 n，判定の正確度を示す自然数パラメタ k

出力: composite を出力したら必ず合成数，probably prime を出力したら高い確率で素数

n が偶数ならば，2 の時 probably prime を，それ以外ならば composite を出力する．

n が奇数ならば，$n-1 = b2^d$ と分解して，以下を k 回繰り返す．

(1) $[2, n-1]$ から a をランダムに選ぶ．

(2) $a^b \not\equiv 1$ かつ $\forall i \leq d-1, a^{b2^i} \not\equiv -1$，または，$a^{n-1} \not\equiv 1$，であれば composite を出力する．

probably prime を出力する．

このような乱数を使えば多項式時間で計算可能な問題のクラスとして，BPP や RP などが知られている．これらが真に P と異なるかどうかは分かっていない．もし，乱数に見える数列が計算可能に作れるのであれば，これらは P と一致する．

11.2 Random Walk SAT

最も単純な 1 次元ランダムウォークは，ある点からスタートし，確率 $\frac{1}{2}$ で右か左に移動する．今，ある整数 N を固定し，$x = n(0 \leq n \leq N)$ からランダムウォークをスタートさせ，$x = N$ にたどり着く確率を求めたい．話を単純にするために，$x = 0$ または $x = N$ にたどり着いたらそこで終了するとし，$x = N$ にたどり着く確率を p_n とする．明らかに $p_0 = 0, p_N = 1$ である．また，漸化式，

$$p_n = \frac{1}{2}p_{n-1} + \frac{1}{2}p_{n+1}$$

が成り立つ．

$$p_{n+1} - p_n = p_n - p_{n-1}$$

と変形することで，

$$p_n = \frac{n}{N}$$

が分かる．$N \to \infty$ を考えれば，ランダムウォークは確率 1 で最初の点からいくらでも離れることが分かる．これは，ギャンブラーの破産問題と言われている．

これを利用して SAT 問題を解こう．まず，論理式を変形して，

$$(...) \wedge (...) \wedge (...)$$

という形に直す．各 () 内には $\vee$ と $\neg$ しか含まれていないようにする．これが充足可能であるためには，各節が T である必要がある．適当に真理値を割り当てるところから始める．すると，各節は T や F である．F となっている節をランダムに選び，そこに現れる変数をランダムに選んで，真偽値を変更する．これを繰り返す．T となる節の個数はランダムウォークのような振る舞いをし，全探索するよりも平均的にははるかに早く充足割当に到達する．

11.3 BPP

様々なものが 2 進有限列で表現される．文字であれば ascii コードや UTF-8 など．色であれば BMP，jpeg, png など．音であれば wave，mp3 など．計算の側面を考えるときには，2 進有限列だけを考えれば良い．

$A \subseteq 2^{<\omega}$ を**決定問題** (decision problem) と呼ぶ．自然数だと思うならば「素数の集合」は決定問題の 1 つである．グラフだと思えば「連結なグラフ」は決定問題の 1 つである．

問題 A に対して，多項式時間計算可能な関数 $f : 2^{<\omega} \to 2$ が存在して，

$$f(\sigma) = 1 \iff \sigma \in A$$

であるとき，A は多項式時間計算可能であるといい，$A \in \mathrm{P}$ で表す．P は多項式時間で計算可能な問題の集合である．多項式時間計算可能であるとは，$f(\sigma)$ の計算時間が，適当な多項式 p に対して，$p(|\sigma|)$ で抑えられることをいう．

問題 A に対して，次のような多項式時間計算可能な関数 $f : 2^{<\omega} \times 2^{<\omega} \to 2$ が存在するとき，$A \in \mathrm{NP}$ と書く．

(1) すべての $x \in A$ に対して，ある $y \in 2^{<\omega}$ が存在して，$f(x, y) = 1$.
(2) すべての $x \notin A$ に対して，すべての $y \in < \omega$ に対して，$f(x, y) = 0$.

$\mathrm{P} \subseteq \mathrm{NP}$ である．単に y を使わない多項式時間計算可能関数を考えれば良い．

問題 $A \in \mathrm{BPP}$ であるとは，次の多項式時間計算可能関数 $f : 2^{<\omega} \times 2^{<\omega} \to 2$ と多項式 p が存在することをいう．

(1) すべての $x \in A$ に対して，$f(x, y) = 1$ となる $y \in 2^{p(|x|)}$ が 2/3 以上.
(2) すべての $x \notin A$ に対して，$f(x, y) = 0$ となる $y \in 2^{p(|x|)}$ が 2/3 以上.

ここで，2/3 という数に深い意味はない．1/2 よりも大きければ，何回も繰り返して多数派を取ることで (これは多項式時間計算可能)，いくらでも確率を高くすることができる．

乱数とは「ある確率分布の実現系列と区別がつかないもの」であるが，すべての計算可能な関数で区別がつかないものは，圧縮不可能な列であり，計算可能ではなくなってしまう．現実的に乱数に見える列を作りたいという欲求は叶えられなくなってしまう．

そこで，「素早い計算では区別がつかない列」として擬似乱数を定義したい．そのような列が素早く作れるかが問題となる．これを深く理解するために，一方向関数という概念を導入しよう．

11.4 一方向関数

一方向関数とは，評価することは簡単だが，逆の計算は難しいような関数を言う．例えば，パズルなどは，答えを聞けばすぐ分かるが，答えを見つけるには時間がかかる．2 つの数の掛け算は簡単だが，素因数分解には時間がかかる．

定義 11.4. 関数 $f : 2^{<\omega} \to 2^{<\omega}$ が**一方向関数**であるとは，以下の 2 条件を満たすことをいう．

(1) f は多項式時間計算可能．

(2) すべての多項式時間乱択アルゴリズム A と多項式 P について，ほとんどすべての n で，

$$\Pr_{x\in 2^n}[A(f(x), 1^n) \in f^{-1}(f(x))] < \frac{1}{p(n)}$$

ここで確率は $x \in 2^n$ の一様分布と乱択で使われる乱数に関するものである．

逆の計算が難しいということをどのように定式化されているのかを見ておこう．$f(x) = x$ という関数は一方向関数ではない．入力の桁数 n と値 $f(x)$ が分かっている時に，元の入力 x が多項式時間で計算できるのであれば，そのアルゴリズムを A とすれば，

$$A(f(x), 1^n) = x \in f^{-1}(f(x))$$

なので，この確率は 1 である．このときには f は一方向関数ではない．

関数 f は，入力 x に対し，最後の bit を 0 に変更するものとしよう．このとき，$f(x)$ から x は復元できない．$A(f(x), 1^n) = f(x)$ とすれば，これは多項式時間計算可能で，確率 $\frac{1}{2}$ で正しく判定できる．なので，一方向関数ではない．

f がとても難しいものであると，$f(x)$ と n という情報だけでは，x を推測することはできなくなる．当てずっぽうで出力すれば，正解する確率は 2^{-n} である．これは任意の多項式時間 $p(n)$ に対して，$1/p(n)$ よりも早く小さくなる．

2 つのものが異なるということは，それらが区別できることを言う．この確率が 0 と異なることを言うには，多項式時間で区別できて欲しいが，一方向関数は，それができないことを意味している．

11.5 ハードコア述語

$x \in 2^n$ の半分の桁数が分かったとしても，正解する確率は $2^{-n/2}$ であり，これも $1/p(n)$ よりも早く小さくなる．よって，f が一方向関数であれば，$f'(x, r) = (f(x), r)$ もまた一方向関数である．一方向関数であるということは，逆計算を行った時に，x の n 桁のうち推測が難しい桁がある程度存在するということを意味している．このような部分を**ハードコア**という．

定義 11.5. 多項式時間計算可能述語 $b : 2^{<\omega} \to 2$ が f の**ハードコア述語**であるとは，すべての多項式時間乱択アルゴリズム A と正の多項式 p について，十分大きな n に対し，

$$\Pr[A(f(x)) = b(x)] < \frac{1}{2} + \frac{1}{p(n)}$$

すなわち x における情報 b は，$f(x)$ からは計算出来ないことを意味している．

定理 11.6. f を一方向関数とする．$b(x, r) = \sum_n x_i r_i$ は，$f'(x, r) = (f(x), r)$ のハードコア述語となる．

証明のアイディアとしては，b がハードコア述語でなければ，b を高い確率で計算するアルゴリズム B が存在して，b に適当な r を入れることで x_i が計算できることから，B を使って高い確率で x_i を計算できる．それぞれの桁数でできるから，f の逆計算ができる．

11.6　擬似乱数生成器

定義 11.7. 決定的な多項式時間アルゴリズム G が**擬似乱数生成器**であるとは，伸長関数 $l : \mathbb{N} \to \mathbb{N}$ が存在して，任意の多項式時間乱択アルゴリズム D と正の多項式 p について，十分大きな n に対し，

$$|\Pr[D(G(U_k)) = 1] - \Pr[D(U_{l(k)} = 1]| < \frac{1}{p(k)}$$

D は長さ $l(k)$ の文字列を 0 と 1 に分けるものであるとしよう．$D(G(U_k))$ は長さ K の文字列に対する長さ $l(k)$ の乱数に対しても，ほぼ割合で 0 と 1 を分けることを意味している．

$l(k) > k$ という条件があるが，$l(k)$ としては長いほど優秀なアルゴリズムということになる．実は $l(k) = k + 1$ の擬似乱数生成器を作れれば，それを繰り返すことで任意の長さの擬似乱数生成器を作れるので，$l(k) = k + 1$ の場合が作れれば十分である．

定理 11.8. 一方向関数が存在するとき，またその時に限り，擬似乱数生成器は存在する．

$l(k) = 2k$ となる擬似乱数生成器 G が存在したとすると，$x, y \in 2^k$ に対して，$f(x, y) = G(x)$ とすれば，f は多項式時間計算可能で，長さ保存．また，f は一方向関数である．そうでなければ，$G(x)$ から x が逆計算可能となり，G は擬似乱数生成器ではないことが示せる．

逆方向の証明はとても難しい．少し強い仮定として，f が 1-1，長さ保存のいち方向関数であれば，擬似乱数生成器となる．b をハードコア述語として，$G(s) = f(s)b(s)$ とすれば擬似乱数生成器となるから．

12　第 13 回　科学哲学

科学とは何か．何が目的なのか．その科学の中で，ランダム性が担う役割を見ていこう．

科学という言葉には，

(1) 科学的な手段
(2) 科学的な手段を用いて得られた理論
(3) 科学の理論を元に作られた技術

など様々なものを指すことがある．ここでは，科学的な手段や科学の理論に注目する．

ごく基本的なモデルとして，

実験（データ）→説明（理論）→予測

というものを考える．

科学の出発点は実験であり，常に観察から始まる．宗教では聖書など，その宗教の創始者の言葉を出発点にすることが多いことと対照的である．例えば，ものを持って手を離したら下に移動するなどの観察が出発点である．

その観察を説明するのが理論である．なぜそのような現象が起こるのかを説明する．古来から多くの人が様々な意見を出してきた．例えば，「人間にとって家は最も心地よく落ち着くところであるから，一度家から出かけたとしてもやがて家に戻ってくる．ちょうどそのように，物にとって最も落ち着くのは地面であるから，地面に帰ろうとするのだ.」これはものが下に移動することを説明する 1 つの理論である．

もし複数の理論があったときに，それらの良し悪しをどのように判定したら良いだろうか．素朴で重要な基準の 1 つが，反証可能性である．カール・ポパーが提唱した．反証可能性とは，その理論が間違っている場合に，その証拠を上げることができるか，ということである．間違っていることを示す方法がない仮説は科学ではない，という言い方もできる．

また，数値で表現することが重要とされた．「下に移動する」という定性的な表現ではなく，「下にどういう加速でどのように移動する」という定量的な表現を取ることが重要である．そのため，理論もまた数字で表現される．それぞれの現象を数学的に理解する数理モデルという考え方が発達する．数学が科学において重要な位置を占める理由である．数値で表現されれば，必然的に予測が正しいかどうか，誤差が少ないかどうかという判定もできる．反証可能性の検証がし易いことも長所の 1 つである．

科学理論のこのような性質上，特定の科学理論が正しいと証明できないことに注意しよう．今のところ，反証は出来ていないというだけのことである．多くの人が反証を試みて，反証できていないという事実が，その科学理論を信頼に値するものにしているという関係を理解してもらいたい．

理論の選択基準として，古くから知られる最もそうな 2 つの考え方を紹介しよう．紀元前のギリシャの哲学者エピクロス (Epikouros) によるもの．「複数の理論があり，それらが反証されていないのであれば，互いに矛盾していたとしても，どちらも保持せよ.」反証可能性を尊重すれば，どちらも拒否できないのだから，どちらも保持するしかない．そのうち，どちらかが反証されるだろう，という考え方．

もう 1 つは，オッカムの剃刀と言われる．「必要のない仮定をつけてはならない.」伊勢田先生は外力がない場合「神が等速に動かしている」「等速で直進する」の 2 つでは，神という余分な仮定がつけられている．必要がないならばはずしなさい．基本的には，理論の仮定は単純な方が良いとされる．ちなみに，オッカムのカミソリを提唱したのは，オッカム村のウィリアムと呼ばれる人である．

科学理論の発展とは反証の歴史であり，統一の歴史である．様々な理論が間違いとされ，様々な理論がより単純な仮定からの帰結として理解されるようになっていく．例えばニュートン力学は，地上の力学と天上の力学の統一として理解できる．

悪魔の証明という言葉でよく知られているように，否定する人に反証する義務はない．証明すべきなのは主張する側である．ところが，科学はその性質上，証明することができないものである．では科学には何ができるのか．

ヘンペルのカラスと呼ばれる話がある．「すべてのカラスは黒い」という命題を証明するためにはどうすればよい

か.「すべての黒くないものはカラスではない」ことを示せば良い．すると，カラスを一羽も調べることなく，正しいことを証明できる．これは非常に直感に反する．かといって，カラスをどれだけ調べても「すべてのカラスは黒い」ことは示せない．ならば，科学で何かを主張するということは根本的にできないことなのだろうか．

確率が高いというところで妥協してはどうか．n 話のカラスを調べてすべて黒かった場合に，次のカラスが黒い確率はどのくらいだと考えるのが良いのだろうか．この場合，カラスが黒かったとして，もしくは白かったとして，その推論は正しかったと言えるのだろうか．言えないのであれば，反証可能性を満たしているとは言えないのではないか．

確率的にしか議論できない場合に，科学的に取るべき態度はどのようなものなのか．現在では確率モデルという考え方が一般的である．数理モデルという考え方を拡張して，確率的に振る舞うと仮定するのである．その反証可能性は，統計的仮説検定により行う．十分なデータが取れるときには，うまく機能する．

今までは理論は「人間がつくる」となんとなく仮定してきた．ところで，理論はどのように作ればよいのか．人工知能，機械学習が注目される現代において，どうやって理論を作ることを計算機に教えたら良いだろうか．

13　第 14 回　科学におけるランダム性

今日は Solomonoff のアルゴリズム的確率の紹介をする．Ray Solomonoff(1926-2009) は初期の人工知能研究の第一人者である．

今日の人工知能は弱い人工知能と言われ，統計的推論や機械学習などをもとに，特定の作業を行う．前回話したモデル (理論) の候補を人間が予め与えておいて，計算機がその中から良い候補を探すという方法を取っている．多くの場合，何が良いかについても答えが決まっている．

強い人工知能は，汎用人工知能とも呼ばれ，具体的な作業が決まっているわけではない中で，何をしたら良いのかを考えることのできるもののことである．将来的にはそういう人工知能を作ることを目標に，日々研究が行われている．汎用人工知能は，これまでの機械学習の延長では難しいと考えられていて，根本的にどのように考えたらよいのかから，考え始める必要がある．そのような文脈で注目されているのが Solomonoff のアルゴリズム的確率の理論である．

前回話たように，問題はどのように理論を見つけるかという問題である．モデル (理論) の集合を与えておいて，その中から良さそうなものを選びなさい，というのが標準的な方法である．しかし，このような方法では，未知のことへの対処ができない．想定していない状況が起こったときにも，これまでの考え方から類推して，適切な判断をする必要がある．そのためには，モデル (理論) の候補を自ら適切に広げ，選択できるようになる必要がある．

Solomonoff のアイデアは，モデルとプログラムを同一視するというものだった．モデルには決定的なモデルも確率的なモデルも含む．モデルがどんな形をしていても，重要なのは入力と出力である．データと予測である．その関係が計算可能でなければ，モデルとして意味がない．だから計算可能なモデルだけを考えよう．そうであれば，モデルとはプログラムのことであるとと思って問題ない．

プログラムと同一視することで，すべてのモデルを順次確認できる．なぜなら Turing による万能機械が存在するので，すべてのプログラムを模倣する機械があるからだ．すべてのモデルの中で，短いプログラムのものは可能性が高いと思い，長いプログラムのものは可能性が低いと思うことにしよう．Epikouros の考えとオッカムの剃刀を，プログラムの長さを通してうまく統一する．こうして得られるのが Solomonoff のアルゴリズム的確率である．

$$M(x) = \sum_{p:U(p)=x*} 2^{-|p|}$$

あとは，σ が与えられたときに，次が i である確率を，ベイズ流に

$$\frac{M(xi)}{M(x)}$$

とする．

驚くべきことに，この予測がすべての計算可能に近似できる予測の中で，漸近的に最も良い予測であることを証明できる．もう少し細かく見てみると，予測するとはデータがランダムとなるモデルを探すことであると言い換えることができる．科学におけるモデルはプログラムと同一視される．発見とはプログラムの発見であり，説明とはプログラムの伝達である．このように計算可能という概念を入れることにより，科学の概念がスッキリと説明できる．ランダム性とは「今のところ規則が見つけられていない」ことを意味している．科学におけるランダム性は，科学のダイナミックな発展を記述する概念でもある．

この Solomonoff のアイデアを実際の世界に応用するにはまだまだ研究が必要である．理論段階から応用・製品への架け橋にはもう少しかかるかもしれない．

13.1　まとめ

この講義では乱数について話をしてきた．乱数の定義，乱数の性質，乱数の検定，乱数の作り方，乱数の使い方，などを見てきた．あまり着目されない概念だが，現実社会においてとても重要な役割を果たしていることが分かってもらえればありがたい．乱数をどのように理解したら良いのか，まだ未解決の問題が多く，進歩の速い分野でもある．理論と応用，学問と社会，理想と現実の境界で試行錯誤する研究者の姿が少しでも伝わっていると良いと思う．