通信の効率化の数学 — ネットワーク符号化

河東　泰之

東京大学大学院数理科学研究科

2011 年 11 月 9 日

符号と数学 — インターネット時代の数学

インターネット，携帯電話などの普及により，ネットワークを行き交うデータ量は爆発的に増えた．文章，音楽，画像などすべてのデータはデジタル化して数字として送られる．たとえばビデオをダウンロードする際のデータ量は莫大であり，今後その勢いは加速する一方であろう．

大量のデータを効率的かつ安全に送るには，もちろん機械についての工学的な技術の発展が必要であるが，そのほかにさまざまな数学的工夫が必要であり，その裏には実用性に乏しいと思われていたかなり「高級」な数学が使われているものもある．

その理論的背景について 3 回に分けて解説する．

1 回目の講義では「効率」について説明する．簡単なモデルをもとに，データ転送の効率向上ついて考える．ここで扱う「ネットワーク符号化」とは 21 世紀に入ってから考えられた手法で，大規模な実用化を前に様々な研究が現在進めらているものである．

2 回目の講義では「安全」について説明する．インターネット上で，クレジットカード番号など，さまざまな個人情報が送られている．そのような処理を安全に行うための数学的工夫について述べる．

以上のテーマの実用的重要性は明らかであるが，数学の理論的側面からは「深遠さ」に欠けるように思う人もいる．そこで，このようなテーマが現代数学の最先端，フィールズ賞にも結び付くようなテーマと密接に関係していることを 3 回目の講義で示す．

私のバックグラウンド – なぜこのような話題を取り上げるか

日本初の個人用コンピュータ TK-80（1976 年発売），さらに Apple II を中学，高校時代に使用.
大学生時代に ASCII 社で著書 3 冊，ゲームプログラム集 2 本，月刊誌連載を含む記事多数を執筆.

ネットワーク通信工学

[現在の専門] 無限次元空間の数理物理学 – 理論物理学に関連する数学的構造の研究（抽象純粋数学）.
現在のデジタルコンピュータの基礎を築いた フォン・ノイマンが創始した理論.

コンピュータネットワークを流れるデータはすべて数字，具体的には 2 進数で表される．すなわち使える数字は 0 と 1 だけである．たとえば広く使われている ASCII コードでは，“A” という文字は 01000001 という 8 桁の 2 進数で表される．(8 桁の 2 進数で，$2^8 = 256$ 通りのデータが送れるので，英字，数字，記号には十分であり，通常 7 桁分しか使わない．) さらに漢字は通常 16 桁の 2 進数で表される．$2^{16} = 65536$ 通りの文字が送れる．

こういった「数字」のかたまりがネットワーク上を送るべきデータである．ある点から別の点へこのような数字を一つ送るということがここで考えている「通信」の最小単位である．実際に数字がどのような電気信号で送られるかは問題としない．

これを次のように簡単な図で表す．

Figure: 通信の最小単位

上図の黒丸を「中継点」，矢印を「経路」と呼ぼう．左側の中継点から右側の中継点に経路がつながっているということがポイントであり，上図の中の点の大きさ，矢印の長さなどは今問題にしていない．

このようなものがたくさん絡み合ったものが通信ネットワークである．インターネットなどでは，中継点の数はとてつもなく大きい．

さてここで送られる「数字」についてもう少し考えてみる．

自然数，実数などはみなよく知っているとおりであるが，実際の通信経路では，無限桁の実数とか，大きさ無制限の自然数などを送ることは不可能である．

実際に一つの単位で送れるものは決まったサイズの整数であり，たとえば「8 桁の 2 進数」がよく単位として使われる．これが**バイト**と呼ばれるものであり，その 1,000 倍がキロバイト，さらにその 1,000 倍がメガバイトである．(正確には $2^{10} = 1024$ 倍だが．)

これらの数は足したり引いたりはできる．(9 桁目にあふれたものは捨てる．) しかし，**掛け算，割り算**もしたい．そこで次のように考える．

整数全体の集合を $\mathbb{Z}$ と書く．自然数 p を一つ決め，整数を p で割った余りを考える．余りは $0, 1, \ldots, p-1$ なのでこれらの数について，足し算，掛け算を，普通に足したり掛けたりした後，p で割った余りで考えることにする．例えば $3 + (p-1) = p + 2$ を p で割った余りは 2 なので，$3 + (p-1) = 2$ と書く．
($3 + (p-1) = 2 \bmod p$ のような書き方の方がなじみがあるかもしれないが，単に等号で書く．) 同様に $2 \times (p-2) = 2p - 4$ のことを $2 \times (p-2) = p - 4$ と書く．($p \geq 4$ とした．) この集合を $\mathbb{Z}/p\mathbb{Z}$ と書く．

足し算，掛け算については通常の交換法則，結合法則，分配法則などが成り立つ．たとえば $a(b+c) = ab + ac$ などである．

しかしたとえば $p = 6$ とすると．2×3 を 6 で割った余りは 0 なので，$2 \times 3 = 0$ となってしまう．これは普通の数と違う規則であり，たとえば「2 で割る」ということはできないことになる．(もしできたら両辺を 2 で割って，$3 = 0$ となってしまうがこれはもちろん正しくない．)

そこで今度は p を素数としてみよう．$p \neq 1$ であって，p を割り切る自然数は 1 と p だけということである．$2, 3, 5, 7, 11, \ldots$ などがその例である．ここで $xy = 0$ とすると，xy が p で割り切れると言うことだが，p が素数であることから，x, y のいずれかが p の倍数であることになる．すなわち，$x = 0$ または $y = 0$ ということである．これは普通の数と同じ規則である．

さらに実は，p が素数であれば，0 でない x についていつでも，$xy = 1$ となる y があることが示せる．たとえば $p = 7$ のとき，$2 \times 4 = 3 \times 5 = 6 \times 6 = 1$ である．

このとき $ax = b$ であれば両辺に y をかけて，$axy = by$ となるが左辺で $xy = 1$ となるので，$a = by$ が得られる．これは $ax = b$ の両辺を x で割ったことになっている．このように，p が素数であると，p で割った余りの世界で，割り算ができることになる．

割り算ができると，普通の数とほとんど同じように扱うことができる．(ただし大小比較はできない．) このような集合を**体** (タイ) と言う．$\{0, 1, 2, \ldots, p-1\}$ は有限集合なので，**有限体**と言う．

さて，8 桁の 2 進数は 10 進数では 0 から 255 までである．これらは 256 で割った余りと思うのが，コンピュータ上の計算と相性が良い．256 が素数であれば，これまでの話とうまく合うが，もちろん $2^8 = 256$ は素数ではない．

そこで突然だが $x^4 + 3x^2 - 5x + 7$ のような多項式を考える．多項式も整数と同じように，足したり引いたり掛けたりできる (が，割り算はぴったり割り切れるときしかできない．)

割り切れないときは，多項式で割り算すると，余りが出る．普通の整数のときは，余りは割る数より小さいが，多項式の時は余りは次数が割る多項式より低くなる．これらは大変よく似ている．

多項式でも「素数」に似たものがあり，それで割った余りを考えるとよい．たとえば，実数を係数とする多項式を考え，x^2+1 で割った余りを考える．余りはいつでも $ax+b$ (a,b は実数) の形である．掛け算は

$$\begin{aligned}(ax+b)(cx+d) &= acx^2+(ad+bc)x+bd \\ &= ac(x^2+1)+(ad+bc)x+bd-ac \\ &= (ad+bc)x+bd-ac\end{aligned}$$

と計算される．この規則より，$a=b=0$ でなければ

$$(ax+b)\left(\frac{-a}{a^2+b^2}x+\frac{b}{a^2+b^2}\right)=1$$

であるので割り算ができる．

さらに係数も「素数で割った余り」を使うことができる．たとえば「2 で割った余り」を考えると数は，$0, 1$ の 2 種類だけである．今度は $x^2 + x + 1$ で割った余りを考えると，余りはいつでも $ax + b$ $(a, b = 0, 1)$ の形であり，掛け算は

$$\begin{aligned}(ax + b)(cx + d) &= acx^2 + (ad + bc)x + bd \\ &= ac(x^2 + x + 1) + (ad + bc - ac)x \\ &\qquad + bd - ac \\ &= (ad + bc - ac)x + bd - ac\end{aligned}$$

と計算される．この世界では 0 でない多項式は $1, x, x + 1$ の 3 つだけであり，$x(x + 1) = (x + 1)x = 1$ なので割り算ができる．

$\{0, 1, x, x+1\}$ は有限体をなしており，4 つの要素からなる．2 次式で割ったため，余りは 1 次式となり，$x, 1$ の係数がそれぞれ 2 個ずつなので，$2^2 = 4$ 通りとなった．

この方法の拡張により，2^n 個の要素を持つ有限体が作れ，コンピュータ処理に適している．すなわち，適切な n 次式をとり，係数は $0, 1$ のみ (2 で割った余りを考える) とすると，余りは $n-1$ 次式で，$x^{n-1}, x^{n-2}, \ldots, 1$ の係数がそれぞれ 2 通りなので，2^n 通りの余りがありうる．

特に $n = 8, 16$ など取るとバイト単位の処理に適しており，パソコン等でも簡単にプログラムが書ける．

通信ネットワーク — **中継点と経路よりなる．**

中継点を黒丸 (「頂点」)，経路を矢印 (「辺」) で表す．

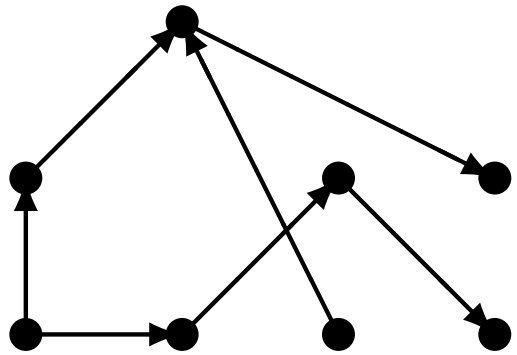

Figure: グラフ

このようなものをグラフという．(「関数のグラフ」とは意味が違う．) どの点とどの点がどの向きにつながっているかだけを考えて，他の情報は捨象している．(ここでは矢印の「長さ」は考えていない．これを考えに入れることもある．)

二つの頂点間に辺が複数あったり，頂点から自分自身へ辺があったりすることもある．今は頂点や辺の数は有限とするが，数学的には無限のものも考えられる．

このような設定を抽象的に考えたものがグラフ理論である．今回の講義ではグラフ理論の応用は示さないが，たとえば，ある頂点から別の頂点へ，辺たちがつながっているか，つながっているとき経路は何本あるか，一番短い経路はどれか，さらに，経路のうち何本か切れても，残りのグラフである頂点から別の頂点へつながっているか，などは重要な問題である．

また通信ネットワークの場合は，データの発信点と終点があるのが普通である．これらはそれぞれ複数あってもよい．

ここで少しわき道にそれるが，グラフ理論の有名な問題である 4 色問題についてふれる．

地図の各国を塗り分けることを考え，隣り合った国同士は違う色で塗ることにする．(ただし，1 点で接している場合は「隣り合った」とは考えない．) この条件のもとで，どんな地図でも 4 色で塗り分けられるか，というのが問題である．

これは次のようにグラフの問題に言い換えられる．

地図が与えられたとする．まず各国を頂点で表す．国 A と国 B が接しているとき，対応する頂点同士を両方向の辺で結ぶ．課題は各頂点に $1, 2, 3, 4$ の番号を振り，辺でつながっている頂点同士は違う数字がふられるようにすることである．

この問題は 100 年以上未解決であったが，1976 年に Appel と Haken によって，答えがイエスであることが示された．約 2,000 パターンの地図をチェックすればよい，ということをまず示し，それらをコンピュータで全部調べたのであった．

場合分けの数が 10 通りくらいで，それを人手でチェックするのなら数学の通常の証明だが，2,000 通りは手ではチェックできないくらい多いので，当時大きな話題になった．

現在に至ってもこれと本質的に異なる証明は知られていない．この問題は現代数学のもっと抽象的な設定の問題に言い換えることができ，もっと簡単な別証明があるかもしれないと期待されているが，まだ見つかっていない．

さて通信ネットワークの話に戻る．実際の通信においては様々な具体的問題がある．

現実のネットワークでは，各中継点での中継には時間がかかる．巨大なネットワークの場合，この時間は無視できないが，数学的に単純な状況を考える場合はこれを無視する．

また，各中継点で受け取ったデータを発信する際にエラーが生じる可能性がある．それどころか単にデータが失われてしまうこともある．また，特定の通信経路に障害が生じて使えなくなることもある．これらはすべて数学的モデルで扱うことができるが，話を簡単にするため，今日は考えない．エラーの対策については 3 回目の講義で取り上げる．

さて今日の本題に入る．通信のもっとも簡単なモデルは，各中継点では，受け取ったデータをそのまま次の中継点に転送すると言うものであり，現実のネットワークでもそうなっている．

これに対し，各中継点で何らかのデータ加工を行い，新しくできたデータを送信するというのが，今日の講義の題目にあるネットワーク符号化である．これは比較的新しい考え方で，2003 年に Li, Yeung, Cai の 3 人によって導入された．

基本的な考え方は，中継点でのデータ処理により，効率的なデータ通信を実現するというものである．

具体的な例で，送るデータは「2 で割った余り」すなわち $0, 1$ とする．足し算は $0+0=1+1=0, 0+1=1+0=1$ である．

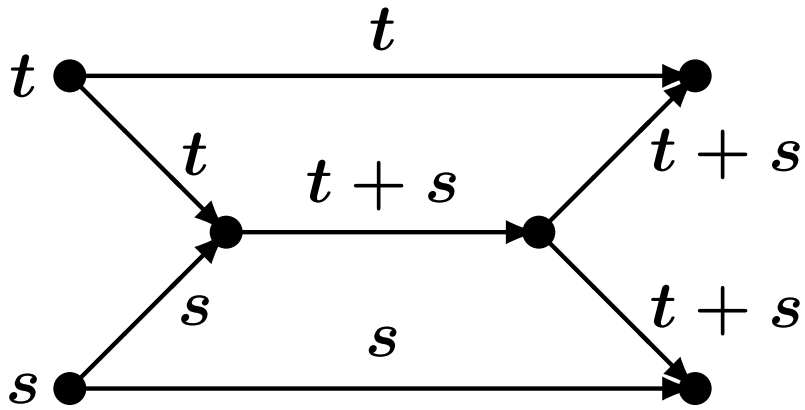

Figure: バタフライ・ネットワーク

1. **左上段から，$t=0,1$, 左下段から $s=0,1$ のデータを右に送り，右上段，右下段で受信する．(中段の 2 点は中継点．)**
2. **左中段で，t,s を受信し，$t+s$ を発信する．**
3. **右上段で，$t, t+s$ を受信するので，t,s がわかる．**
4. **右下段で，$s, t+s$ を受信するので，t,s がわかる．**

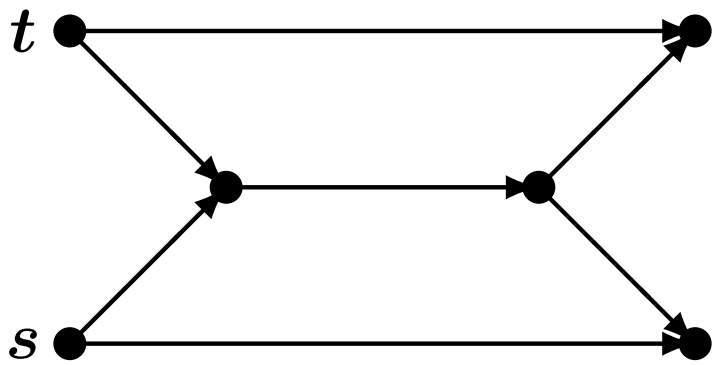

Figure: バタフライ・ネットワーク

1. 左上段から，$t = 0, 1$, 左下段から $s = 0, 1$ のデータを右に送り，右上段，右下段で受信する．(中段の 2 点は中継点．)
2. 各中継点では受け取ったものをそのまま転送する．
3. 中段右矢印にデータ t, s のどちらかが流れる．仮に t とする．
4. 右上段で t, s の両方のデータを得るためには，上段右矢印ではデータ s が流れないといけないが，不可能．

さて実用的な観点からこのアイディアの問題点を検討しよう．

「$t + s$ を送る」などのデータ処理の方法はグラフの形によって決まる．どのようなグラフの時，どのようにデータ処理をすればよいか，という一般的な理論はあるが，現実の大規模ネットワークでそれを計算することは困難である．さらにまた現実のネットワークを表すグラフは，通信障害や新規中継点の誕生によって絶えず変化している．ある時点におけるネットワーク全体のつながり方は誰にもよくわかっていないことが普通である．この立場からは，上の例のようなデータ操作は現実的ではないと言える．

そこを工夫するのが次のアイディアであるが，その前に少し抽象的な数学の説明を行う．

有限体を一つ選ぶ．たとえば「2 で割った余り」としよう．その数が n 個並んだもの $(x_1, x_2, \ldots, x_n)$ を n 次元ベクトルと思う．各成分が $0, 1$ の 2 通りだから点は全部で 2^n 個しかないが，これを通常の n 次元空間と同様に考える．たとえばこの中の直線，平面などにあたるものを考える．

通常の平面では，方程式 $x + y = 0$ は直線を定め，通常の空間では，方程式 $x + y + z = 0$ は平面を定める．同様に (x_1, x_2) という形の 4 個の点の中で $x_1 + x_2 = 0$ を満たす 2 個の点が「直線」を定め，(x_1, x_2, x_3) という形の 8 個の点の中で $x_1 + x_2 + x_3 = 0$ を満たす 4 個の点が「平面」を定めると考える．このような考え方がとてもうまくいく．

線形代数

このようにベクトルや行列を扱うのが，線形代数と呼ばれる数学であり，世界中の大学で理科系の 1 年生に教えている．(経済学などでもたいへん重要である．)

たとえば，3 次元空間の中に平面が 2 枚あれば，たいていは共通部分は直線である．例外は 2 枚が平行な時，2 枚が一致するときである．このことは直感的にわかりやすいが，線形代数の強力なところは，行列やベクトルの中の数字が実数でなく，有限体の要素であってもよいことである．(あるいは複素数でも同様に O.K. である．)

これによって前ページのように，有限個の点の集まりを直線とか平面とか思ってうまくいくのである．

ランダム・ネットワーク符号化

これは Ho, Koetter, Medard, Karger, Effros によって 2003 年に提唱された手法である．

送るべきデータは「数」であるが，それは有限体の中の数 x と思ってよい．これを n 個並べたベクトル $(x_1, x_2, \ldots, x_n)$ が送りたいデータの単位である．さらにこのベクトルが k 個あるものを送りたいとする．この k 個のベクトルをそのまま送るのではなくて，適当な操作を加えたうえで，ベクトルをばらばらに送る．

中継点を経由した後，最終目的地ではベクトルを k 個またはそれ以上集めて，もとの k 個のベクトルを再現したい．

簡単のため，最終的に送るべきデータを 2 つの n 次元ベクトル $\vec{x}, \vec{y}$ とする．適当な係数 a, b と $a\vec{x} + b\vec{y}$ の組 $(a, b, a\vec{x} + b\vec{y})$ を送ることにする．このようなデータが二組 $(a, b, \vec{u}), (c, d, \vec{v})$ とあったとしよう．$\vec{u} = a\vec{x} + b\vec{y}, \vec{v} = c\vec{x} + d\vec{y}$ であるから，$ad - bc \neq 0$ であれば $\vec{x} = (d\vec{u} - b\vec{v})/(ad - bc)$, $\vec{y} = (-c\vec{u} + a\vec{v})/(ad - bc)$ と解ける．

係数が「でたらめ」に選ばれていれば「たいていは」$ad - bc \neq 0$ なのでこれで解ける．$ad - bc$ は 2×2 行列の行列式である．

$\vec{x}, \vec{y}$ はベクトルなので，$(a, b, a\vec{x} + b\vec{y})$ を送る方が $(\vec{x}, \vec{y})$ を送るより短いことに注意する．

さらに，途中の中継点で $(a, b, \vec{u}), (c, d, \vec{v})$ という二つのデータを受け取ったとしよう．このとき，k, l をランダムに選び，$(ka + lc, kb + ld, k\vec{u} + l\vec{v})$ を作る．もし $\vec{u} = a\vec{x} + b\vec{y}, \vec{v} = c\vec{x} + d\vec{y}$ であれば $k\vec{u} + l\vec{v} = (ka + lc)\vec{x} + (kb + ld)\vec{y}$ であるので，こうやって新しくできたデータ $(ka + lc, kb + ld, k\vec{u} + l\vec{v})$ も元と同じタイプである．

中継点で受け取るデータが 3 つ以上あっても同様である．これにより，各中継点でランダムに係数を選んで上の操作をしても，状況は変わらず，むしろ係数がランダムに混ざる分，行列式が 0 になる可能性が低くなると言ってよい．

上の例では 2 個のベクトルを考え，$(a, b, \vec{u})$ という短いデータ長で考えたが，実際には $(a_1, a_2, \ldots, a_k, \vec{u})$ というものを考える．

このようなデータが k 個集まったとする．$k = 2$ のときは $ad - bc$ という式が出てきたが，これは 2×2 行列 $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ の「行列式」というものである．一般の k のときは $k \times k$ 行列の行列式というものが出てくる．これはある数を与えるが，その値が 0 でなければ，$ad - bc \neq 0$ の時と同様に解いて，元のデータを復元することができる．

おおざっぱには，この行列式の値は「たいてい 0 ではない」ので解くことができるのである．(うまくいかない確率も見積もれる．)

さらにこの方法は，エラー検出にも適している．前ページでは，データが k 個集まったとしたが，データはコンピュータネットワーク上をあまりコントロールできない状態で流れてくるので，$k+1$ 個以上来ることも十分ありうる．このとき，データ相互間に矛盾がないかどうかはすぐにチェックできる．矛盾があればデータ転送にエラーがあったということである．(なお $k-1$ 個以下のデータでは復元できないことはすぐわかる．)

上の場合には，エラーがあったということしかわからないが，もう少し工夫を加えればエラーを (ある程度) 自動的に訂正することもできる．この種の手法については 3 回目の講義で詳しく扱うが，ここでも有限体の上でベクトルや行列を考える手法が有効である．