信頼・効率を生み出す符号化

情報理工学院 情報工学系
金子 晴彦 研究室

金子　晴彦 准教授　1976年東京都生まれ。東京工業大学大学院情報理工学研究科計算工学専攻博士課程修了。2014年、東京工業大学大学院情報理工学研究科計算工学専攻准教授。2016年、改組により同大学情報理工学院情報工学系准教授。

情報分野の発展はめざましく、また多岐にわたっている。金子研究室はその中で符号化理論を対象にした研究、特にハードウェアに適切な符号化を施すための研究を行なっている。本稿では符号化の基本から入り、その後先生の研究である誤り訂正符号とデータ圧縮について紹介する。また、先生の研究のスタンスについても触れる。

コンピュータと符号化

コンピュータ、タブレット、スマートフォン。現代に生きる私たちの周りには、情報機器が溢れている。かつては1台の大型コンピュータを複数人が使っていたが、今は1人が複数のコンピュータを使うような時代、いわゆるユビキタス社会になりつつある。

しかし、こういった機器の仕組みをきちんと理解して使っている人はごく少数だろう。というのも、現代の情報分野は非常に多岐にわたっており、それらを全て知ることは不可能といっても過言ではないからだ。そんな情報分野の研究の一つに金子研究室が行なっている符号化という分野がある。

研究について紹介する前に、コンピュータがどのように情報処理を行なっているか、基本的なことに触れておく。コンピュータはデータを0と1の集まり、いわゆるデジタル信号として処理している。デジタルとはデータが実数のように連続した数ではなく、飛び飛びの値で表現されているということである。逆にいえば、アナログというのは、データが連続した数で表されているものということになる。

符号化とは、デジタルデータを一定の規則に従って目的に応じた形式のデータに変換することである。具体的な例として、QRコードがある。QRコードとは、URLなどの文字列をある規則のもとで2次元の画像に変換したものだが、この変換は符号化である。他にも圧縮、暗号化などが符号化に当たる。

では、金子研究室の研究について実際に見ていこう。金子研究室は現在主に2つのテーマについて研究を行なっている。まずはそのうちの1つ、誤り訂正符号について見ていく。

図1　エラーのタイプ

上のデータ列がエラー発生前、下のデータ列がエラー発生後のものである。

誤り訂正符号

先ほど述べたように、コンピュータはデータをデジタル信号として処理している。しかしデータの送受信がどのように行われているかというと、結局のところ電圧の高低といったアナログ信号でやり取りされている。つまり、コンピュータは送られてきたアナログな電気信号をデジタル的に処理し、処理されたものをアナログな電気信号として送信しているのだ。

この変換や送信といった一連の流れの中で、電気信号の受信タイミングがずれてしまったり、ノイズが含まれたりといったことが起こると、データのエラーが生じうる。しかし、受信側は送信側のデータを直接見ることはできない。そこで、データを符号化することで、送られてきたデータのエラーを訂正することができるようになる。この手法が誤り訂正符号である。

エラーのタイプは大別すると3つある。反転型、挿入型、そして削除型である。反転型とは、例えば0が1、1が0というように、データ列の値が反転してしまったエラーのことである。また、挿入型、削除型はそれぞれデータ列の一部に余分なデータの挿入、あるいは抜け落ちが生じてしまったエラーのことである(**図1**)。このうち反転型エラーの訂正符号は比較的多く研究されているので、先生は研究の余地が多い挿入・削除型エラーの訂正符号の研究に取り組んでいる。

挿入・削除型エラーを訂正する手法として、考えられるものはいくつかある。

その中の一つにマーカー挿入という手法がある。これはあらかじめ決めておいたデータ列をマーカーとして等間隔に挿入し、受信側でそのずれを検出することで誤り訂正をするというものだ。し

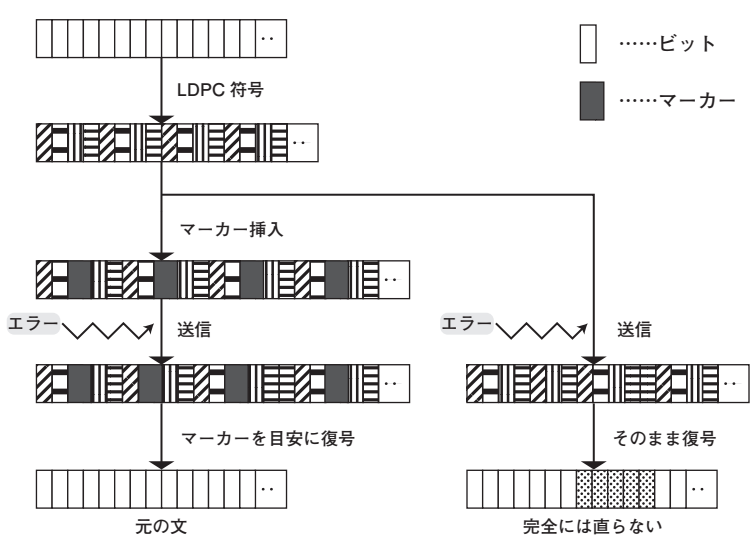

図2　代表的な挿入・削除型誤り訂正符号

左のようにマーカーを挿入することで、それを目安にエラーの位置が推定でき、元の文に復号できる。LDPC符号だけでは右のように完全には直らないことがある。

かし、この手法では大まかなずれしか推定することができない。というのも、この手法ではマーカー間のずれの大きさがわかるだけで、どこにどのようなエラーがあるのかはわからないからだ。

別の手法として、LDPC符号という符号化の手法がある。これは、ある程度小さいエラーであれば正しく復号できる優れた符号である。しかし、長いデータ列での挿入・削除型エラーでは、エラーが発生した後のデータ列が全てずれてしまうため、エラーの大きさがLDPC符号の訂正できる限界よりも大きくなってしまうのだ。

そこで、これらを組み合わせた手法がよく用いられる。まず、LDPC符号でデータ列を符号化し、そのあとマーカーの挿入を行う。こうすることで、データ列の送信時にエラーが発生してしまった場合でも、マーカーで区切った間隔ごとのエラーを大まかに推定してエラーの大きさを細かく分け、さらにLDPC符号の復号を行うことでデータ列のエラーを正確に訂正することができるのだ（**図2**）。

ただ、このような誤り訂正符号は、対象とするハードウェアそれぞれに合うものを考えなければならない。先生が誤り訂正符号の対象としているハードウェアは、BPMという記憶媒体である。従来のハードディスクでは磁性体が連続的に塗られているのに対して、BPMを搭載したハードディスクは磁性体が格子状に区切られている。一見どのようなメリットがあるのかと思ってしまうが、こうすることで、従来のハードディスクと違い磁性体同士の干渉を受けにくくなるため、結果的に記憶密度が上がるのだという。まだ研究段階であるが、将来的な実用化が期待されている。

しかし、このBPMには格子状ゆえの課題もある。従来のハードディスクは、磁性体は連続的に塗ってある。そのためデータを書き込む位置が多少ずれたとしても、読み込んだデータの波形は変わらず、正しく読み出すことができる。ところがBPMは、磁性体間に隙間があることから、1ビット未満の書き込みのずれが原因で波形が大きく変わり、正しい値が読み出せなくなることがある（**図3**）。これまで提案されていた手法はこの問題に対応しておらず、BPMの誤り訂正がうまくいかないことがあるのだ。

そこで先生は、半端なずれに対応した誤り訂正符号について考えている。従来の訂正符号は、復号時にデータ列を1ビットごとに区切ったモデルを想定していたが、エラーの置き方を1ビットずつに限定しないものとし、その上でそのずれが結局どのような結果をもたらすのかについて考察し、シミュレーションを行なった。

現時点では、先生のモデルはまだ実際のものの挙動をうまく表現できていないところがあるので、今はそれを実際のものに近づけている。また、提

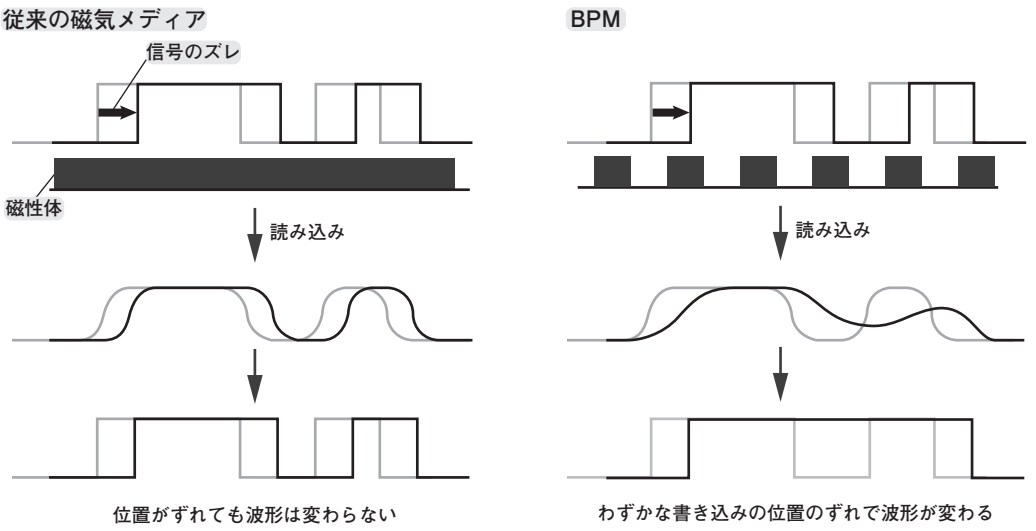

図3　BPM特有の問題

案している手法はあくまで発表されている論文のデータのみに基づいているので、将来的には実際のBPMで実験したいそうだ。

現在実用化されている磁気メディアは最終的なエラー残余の割合が10^{-12}から10^{-15}程度である。その一方で先生の行なっているシミュレーションにおいて、訂正符号によって訂正されたあとのエラーの残余の割合はおよそ5×10^{-5}である。これはシミュレーションにおけるエラーの発生確率が10^{-3}という、比較的エラーが発生しやすい環境を想定していることを考えてもまだまだ高い。先生はこのエラー残余の割合を実用化可能なレベルとされる10^{-7}程度にすべく、研究を重ねている。

データ圧縮

続いて、先生のもう一つの研究テーマである、データ圧縮について見ていく。先生は現在、圧縮を利用してコンピュータの処理の高速化を行う研究をしている。

コンピュータの内部には、プロセッサ、メモリ、ストレージといった情報を処理あるいは記憶する部分がいくつかある。それぞれが決められた処理を実行し、互いにデータをやり取りすることで、コンピュータ全体の処理は行われている。

コンピュータの処理を速くしようとするにあたり、考えられる手法はいくつかある。すぐに思いつくのはCPU、メモリといったコンピュータを構成するパーツの性能をあげることであろう。もちろんこれは重要である。現在までにコンピュータの性能が飛躍的に向上してきているのはこういったパーツ自体の性能が大幅に向上してきたことが主な要因だからだ。

しかしパーツ単体はかなりの高性能化が進んでおり、この方法だけでは限界がある。そのため処理をさらに高速化するにはこれ以外の手法が必要となる。その一つとして、パーツ間のデータ受け渡しの速度をあげるという手法がある。現在のコンピュータの処理において、この部分による遅延は比較的大きいため、ここを改善すればコンピュータの処理全体の高速化につながる。

ここで圧縮技術が利用できる。圧縮で受け渡すデータのサイズを小さくしてやれば、データ受け渡しによる遅延は小さくなるのだ。

圧縮にはいくつか種類があるが、大別するとソフトウェア圧縮とハードウェア圧縮に分かれる。ソフトウェア圧縮は、例えばzipやrarといったものである。これは、データの不必要に長い部分をプログラム上で処理することで圧縮をする。圧縮率は高いのだが、ソフトウェアを介す必要があるため、あまり高速に圧縮できない。そのためコンピュータ内部の処理を速くする目的には使えない。

これに対してハードウェア圧縮は、データの信号を電子回路上で圧縮するものである。これは前者に比べて圧縮の速度が速い。そこで先生は、高速かつ圧縮率の高いハードウェア圧縮のアルゴリズムを作ることで、データパスの速度をあげることを目標としている。

圧縮の一番簡単な手法は、0を省略することだ。一般にデータ列は、そのままだと内容が含まれていない部分が多くある。例えば、コンピュータは整数を1つ表すのに32ビット使うことが多いのだ

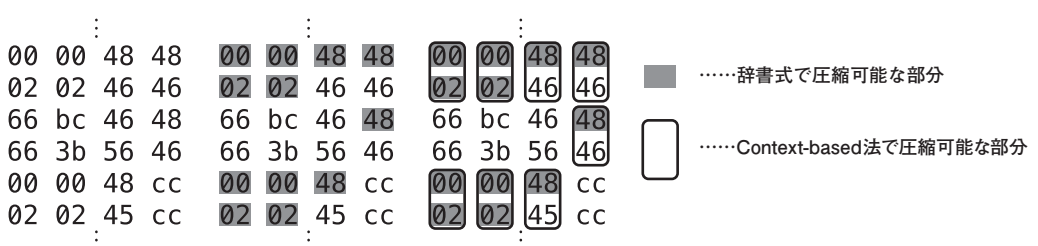

図4　ソフトウェア圧縮の手法

左から、元のデータ列、辞書式による圧縮可能な部分、Context-based法で圧縮可能な部分を表す。

が、32ビット分メモリを確保しておきながら、1などの数を表す際は、結局そのうちの数ビット分しか使わず、残りのビットは全て0になる。こういったデータの中身に関係のない0を省略することでデータを簡単に圧縮できるのだ。

この手法に、辞書式という手法を導入したものもある。これは、出てきたデータ列の一部をしばらく覚えておき、2つ以上同じ部分が出てきたら、1つ目の部分の位置などの情報に変換するという手法である。圧縮速度は少し落ちるが、データ列はより圧縮される。

先生はこれにさらに工夫を加えた。ただ出てきやすいパターンを圧縮するだけでなく、出てきやすいパターンの前後をまとめて短い符号にすることにしたのだ。例えば、(**図4**)のようなデータ列があるとする。ここで比較的多く出てきているパターンは、00、02と48の4回である。従来の手法ではこのよく出てくるパターンを圧縮する。対して先生の提案した手法では、これらのパターン、例えば00について、さらに前後を見る。すると、00の真下には必ず02が来ていることがわかる。そこで、この00と02をまとめて短い文字に変えることで、大幅にデータを圧縮できるのだ。同様に48についても調べてみると、48の下には46が3回、45が1回来ている。そこで、48-46の組み合わせを短い文字に、48-45の組み合わせを少し長い文字に置き換えることで、結果的にデータを圧縮できる。この手法はデータ列の文脈、つまり前後関係を見て符号化するため、Context-based法と呼ばれる。こうすることで、平均的にはデータをさらに圧縮することができる。

ただ現在取り組んでいる手法では、科学技術用の計算といった特殊なデータしか十分に圧縮することができないそうだ。先生はより汎用性のある手法にするため研究を重ねている。

理論とハードウェアの橋渡し

これまでに紹介してきた研究を始め、先生の研究テーマは、誤り訂正や圧縮の技術を利用できる具体的なハードウェアを出発点にすることが多いそうだ。例えば、BPMの誤り訂正符号についての研究はBPMという具体的な研究対象となるメディアが出てきたこと、圧縮についての研究はスーパーコンピュータの研究者たちとコンピュータの高速化について話をし、データ受け渡しのための圧縮について考えたことがきっかけだという。

また、この研究分野はひらめきやアイデアが重要になってくる。先生は積極的にアイデアを話してくれる学生が欲しいと言っていた。ただ、うまくいくアイデアが思いつくまでには、結局試行錯誤が必要だという。先生によるとアイデアが期待通りの成果を出すことは1割程度しかないそうだ。その分、良い圧縮率やエラー残余割合が数字として表れた時の喜びもひとしおなのだ。

最後に、先生に研究の理想について伺ったところ、それは符号化理論とハードウェアの橋渡しをすることのできる研究をすることだという。つまり、単にBPMや、コンピュータといった具体的なハードウェアに合った符号化技術を開発するだけでなく、符号化理論からより効率のよいハードウェア、例えば処理の高速なコンピュータの構造あるいは設計といったものを提案するという研究のスタンスだ。

現代の高度情報社会の発展は、コンピュータシステムの技術革新に支えられているといっても過言ではない。最近話題になっている人工知能、バーチャルリアリティといった研究分野も、こうした基礎研究があったからこそ生まれたという事実がある。これまで紹介してきた先生の研究は、いずれもより高速で信頼のおけるコンピュータシステムの開発につながっており、それはつまり現代の情報社会の発展に必要不可欠なものである。

執筆者より

今回の取材では、符号化という複雑な数学を用いた先生の研究について、わかりやすく理解することができました。

また、研究内容の他にも、符号化の進歩やテーマの探し方といった話もしてくださり、それらはとても興味深いものでした。

お忙しい中快く取材を引き受けてくださった金子先生に心より御礼申し上げます。(小西　優実)