情報処理 (7)

情報の伝達と表現方法

増原 英彦

情報の表現方法は、これまでに見たような 2 進符号化だけとは限らない。ここでは、情報の表現方法を工夫することで、伝達や記憶の過程で発生した誤りを見つける誤り検出・訂正方法や、より少ないビット数で表わす圧縮、また盗聴されても内容が分からないようにする暗号化について紹介する。

1 情報の誤り

ネットワークはデータを電気信号に変えて送るため、信号の減衰や電気的雑音によって送ったデータが正しく届かないことがある。電波や光を用いた通信も同様のことが起こる。また、計算機の主記憶装置 (メモリ) は、微小なコンデンサの帯電によってデータを記録しているため、宇宙線の影響などでデータが変化してしまうこともある。

問題 7–1: (誤りが生じる確率) マイクロン社の資料によると、16M バイトのメモリを平均 16 年間使い続けると 1 ビットのデータに誤りが起きる[1]。512M バイトのメモリを搭載した計算機が 1149 台あった場合、どれか 1 台の計算機で 1 ビットのデータに誤りが起きるのに要する平均の時間はどれだけか? 簡単のためにメモリの量が倍になれば、誤りが起きるのに要する平均時間は半分になるとしてよい。

1.1 誤り検出 (error detection)

通信して送られてきたデータや記憶装置から取り出したデータは常に正しいとは限らないため、データが正しいかどうかを確認する手段が必要になる。このような手段のことを**誤り検出**(error detection) と言う。通信において送られてきたデータの誤り検出が可能であれば、データを再度送り直してもらうことができる。

二重化 最も単純な誤り検出方法は、全てのデータを 2 度送る**二重化**と呼ばれるものである。誤りかどうかを検査するには、2 つのデータが一致するかどうかを確めればよい。もちろん、2 度送られたデータが全く同じように誤っていた場合には、誤りが起きていることを検出できないが、その確率は非常に低い。

誤り検出方法を考える際には、「元の n ビットのデータに対してどれだけ余計なデータを送れば、そのデータの中の何ビットかに誤りが起きたことを知ることができるか」という尺度が用いられる。上の二重化では、1 ビットのデータに 1 ビット付加することで (その 1 ビットに) 誤りが起きたかどうかを検出できる。

[1] 構成や計算方法によって変わる。

パリティ　ほとんどの場合、誤りの起きる確率は非常に低いので、データ量を2倍にしてまで誤りの検出を行う意義は低い。そこで実際には、数〜数十ビットのデータをひとまとめにして、それに対して誤り検出のための数ビットの情報を付加する。

パリティ(偶奇) ビットは、簡単な誤り検出方法の1つである。n ビットのデータがあったとき、その中に「1」が何個現われているかを数え、それが偶数であれば「0」、奇数であれば「1」とする。例えば7ビットにつき1ビットのパリティを付ける場合、「0000000」であれば「1」は0個＝偶数なので「0」を、「0101010」であれば「1」は3個＝奇数なので「1」を付加するといった具合である。パリティも含めた8ビットの中で、どれか1つが誤っている (0と1が逆になっている) 場合、「パリティビットは1の個数の偶奇」であるという関係が成り立たなくなるので、誤りであることが検出できる。ただし検出できたとしても、どこのビットが誤っているかまでは分からない。また、誤りが2つあった場合にはパリティでは誤りがあることを検出できなくなる。

1.2　誤りの訂正 (error correction)

多数決　誤りを検出するだけでなく、その誤りを訂正するような方法もある。例えば、最初に考えた「全てのデータを2度送る」かわりに「全てのデータを3度送る」方法を考えてみよう。この場合、送られた3ビットのうち1ビットが全て同じでないとしても、多数決によって元の正しいデータを復元できる。

ハミング符号　より巧妙な方法として、数ビットのデータに何ビットかの情報を付加することで誤りを訂正する方法もある。そのような方法の中で、よく使われているものには1ビットの誤りを訂正できる**ハミング (Hamming) 符号**というものがある。

次の図は、4ビットの情報に3ビット追加するハミング符号の例である。「0110」という4ビットの情報を送る際には、この表の「付加情報」の部分を加えた「0110011」という7ビットの情報を送る。情報を受け取った側では「最も近い行」を選ぶことで誤りの訂正ができる。例えば「0110011」が誤って「0100011」となって届いたとする。受け手は、表の各行に対して1ビットずつ比較し、異なるビットの個数を数える (表の右端の列)。すると「1個だけ異なる」のは「0110011」だけなので、元の情報は「0110」であったと訂正ができる。

元の情報				付加情報			「0100011」と一致しないビット数
0	0	0	0	0	0	0	3
0	0	0	1	1	1	1	3
0	0	1	0	1	1	0	4
0	0	1	1	0	0	1	4
0	1	0	0	1	0	1	2
0	1	0	1	0	1	0	2
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	5
1	0	0	0	0	1	1	2
1	0	0	1	1	0	0	6
1	0	1	0	1	0	1	5
1	0	1	1	0	1	0	5
1	1	0	0	1	1	0	3
1	1	0	1	0	0	1	3
1	1	1	0	0	0	0	4
1	1	1	1	1	1	1	4

このハミング符号の付加情報にさらに1ビット加えると、1箇所誤っていた場合には元の情報に訂正ができ、2箇所誤っていた場合は訂正はできないが誤っていたことを検出できるようになる。

問題 7–2: (ハミング符号) 上の4ビットに3ビット付加するハミング符号では、2箇所のビットが誤っていたことは検出できない。このことを具体例で示せ。

問題 7–3: (拡大ハミング符号) ハミング符号にもう1ビット付け加えると、2箇所の誤りを検出できるようになる。どのようなビットを加えて、どのように検査すればよいか?

2 情報量—再び

これまで情報量は、「2^n 通りの中から1つ選ばれたという事実は $\log 2^n = n$ ビット」のように定義してきた。この定義は次のように一般化される:

ある場合になる確率が p だったとき、その場合になったという事実の情報量は $\log_2 \frac{1}{p}$ ビット

この定義は、「『ありがちな場合が起きたこと』の情報量は小さく、『稀な場合が起きたこと』の情報量は大きい」と理解できる。一般にニュースの価値も滅多に起きない事ほど価値が高いと言えるが、それと直感的に一致している。

2.1 平均情報量

例えば、ゆがんだ四面体のサイコロがあって、1の目が出る確率は「1/2」、2の目が出る確率は「1/4」、3の目と4の目が出る確率は「1/8」ずつだったとする。このサイコロが発生させる1つ目の情報量は「各目の出る確率に、各目が出たことに対する情報量を掛けたものの和」として定義される。つまりこの場合は、

$$\frac{1}{2} \times \log_2 \frac{1}{1/2} + \frac{1}{4} \times \log_2 \frac{1}{1/4} + \frac{1}{8} \times \log_2 \frac{1}{1/8} + \frac{1}{8} \times \log_2 \frac{1}{1/8} = \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{3}{8} = \frac{7}{4} = 1.75(\text{ビット})$$

となる。単純に1〜4の目が同じ確率で現われる場合は $\log_2 1/4 = 2$ ビットであったので、より少ない情報量しか持っていないことになる。

一般に、n 通りの場合があって、i 番目の場合が起きる確率が p_i であるようなとき、全体を平均した情報量は

$$\sum_{i=1,2,\ldots} p_i \log_2 \frac{1}{p_i}$$

と定義される。この平均情報量は、n 通りの場合を符号化するのに必要なビット数の下限になっている。

問題 7–4: (一般化された情報量) いままでの定義が一般化された情報量の定義と矛盾していないことを示せ。

問題 7–5: (イカサマサイコロ) 六面体のサイコロの目が次のような確率で出るとする。

目	1	2	3	4	5	6
確率	1/4	1/4	1/4	1/8	1/16	1/16

このときの各目の情報量と、このサイコロの目の平均情報量を求めよ。

3 圧縮

3.1 非等長の符号化

上の例に出したサイコロを振って出た目を並べてゆくと例えば「1 2 4 2 1 1 1 2 1 1 4 1 1 3 1 4 4 1 1 ...」のようになる。これを符号化する1つの方法は「1」→「01」「2」→「10」「3」→「11」「4」→「00」のように、それぞれの目を2ビットで符号する方法であった。この方法で符号した場合、20個の目「1 2 4 2 1 1 1 2 1 1 4 1 1 3 1 4 4 1 1」は「01100010010101011001010001011101000000101」と40ビットに符号化される。

しかし、全ての場合を同じ長さにする必然性はない。例えば「1」→「0」「2」→「10」「3」→「110」「4」→「111」のように符号化してみよう。そうすると「1 2 4 2 1 1 1 2 1 1 4 1 1 3 1 4 4 1 1」は「0101111000001000111001100111111100」のように33ビットになる。この符号化された情報から元のサイコロの目の並びを復元することができることに注意せよ。

それぞれの目が出る確率と符号の長さを考えると、1つ目を符号化するのに必要なビット数の平均値は $1\times 1/2 + 2\times 1/4 + 3\times 1/8 + 3\times 1/8 = 1.75$ ビットということになる。(これは丁度、このサイコロの目が持つ平均情報量と一致している。)

このように、符号化の方法を工夫して同じ情報をより短く表現することを情報の**圧縮**と言う。上の例では、単純に符号化した場合 (40ビット) に比べて $1 - 33/40 = 17.5\%$ 圧縮されていることになる。

上の例では、起きる確率が高い場合には短い符号を、低い場合には長い符号を割り当てることで、平均的には短い表現を得ている。このような符号化形式を作る方法には、**ハフマン符号化**などの方法が知られている。例えば英文を符号化する場合には、アルファベットの文字によって使用頻度が違うので、頻度に応じた長さの符号を割り当ててやれば、より短く表現できる。

問題 7–6: (非等長の符号化) 上にならって問題 7–5 のサイコロの各目を符号化する方法を考え、「3 3 5 3 2 3 4 1 2 2 6 3 2 2 5 2 1 5 1 3」という目の並びを符号化せよ。何ビットになるか?

3.2 繰り返しに注目した圧縮

圧縮の方法は、情報の性質によって様々なものが使われる。以下ではアルファベットから成る文章を圧縮する例で説明するが、各アルファベットが「場合」であると考えれば一般的な情報の圧縮となる。

同じ文字が繰り返し現われるような場合を考えよう。いま、アルファベットが「a,b,c,d」の 4 文字しかなく、「ccccccaaaaddddddaacaaabbbbcddddddddddaaa」のように同じ文字が繰り返し現われるとする。この 40 文字の情報を単純に符号化した場合、各文字を 2 ビットで表現するので 80 ビットになる。

ランレングス圧縮　次にこの文字列を「文字」と「繰り返し回数」で表わすことにする。ただし、「繰り返し回数」は 1〜4 までとする。(それ以上繰り返しがある場合には、2 回同じ文字が繰り返しているとする。) すると、「ccccccaaaaddddddaacaaabbbbcddddddddddaaa」は「c4c3a4d4d1a2c1a3b4c1d4d4d2a3」となる。さらに「a」→「00」「b」→「01」「c」→「10」「d」→「11」「1」→「00」「2」→「01」「3」→「10」「4」→「11」のように各々2 ビットの符号を割り当てると、「10111010001111111100000110000010011110001111111111010010」56 ビットで表現できるようになる。このような圧縮を**ランレングス (run length) 圧縮**という。

Lempel-Ziv 圧縮　一般的な文章を符号化することを考えると、同じ文字が続くことはさほど多くないが、数文字単位のまとまりで考えると、繰り返しがあることが多い。例えば

> peter piper picked a peck of pickled peppers.
> a peck of pickled peppers peter piper picked
> if peter piper picked a peck of pickled peppers
> how many pickled peppers did peter piper pick

という詩であれば、同じ単語が何度も出ているだけでなく、「per」「ck」「ed」などの数文字が繰り返されていることも分かるだろう。このような性質に注目して、「手前に現われた文字列が繰り返させる」ことを特別な符号を使って表わすような圧縮方法もある。

その中でもよく使われているのは、Lempel と Ziv が考案した Lempel-Ziv 圧縮法とその改良である。この方法では、何個からの文字の並び (「固まり」と呼ぶ) に番号をつけ、新しい固まりを「前に出てきた固まりの番号」＋1 文字のように表わす。上の詩をこの方法に従って固まりに分けると下のようになる。例えば番号 1 の固まりは、「0 番の固まり (これは無しを表わす) に p を加えたもの」ということで「p」を表わす。番号 2 ン固まりは「無しに e を加えたもの」、番号 4 の固まりは「2 番の固まり (つまり e) に r を加えたもの」となる。

番号	前	追加	番号	前	追加	番号	前	追加	番号	前	追加	番号	前	追加
1	0	p	16	11	k	31	12	l	46	23		61	0	y
2	0	e	17	5	o	32	21	p	47	30	k	62	19	c
3	0	t	18	0	f	33	2	p	48	21	a	63	31	e
4	2	r	19	9	i	34	23	s	49	15	c	64	53	p
5	0		20	16	l	35	15	t	50	27	o	65	33	p
6	1	i	21	13		36	4		51	43	i	66	55	
7	1	e	22	7	p	37	6	p	52	20	e	67	41	i
8	0	r	23	7	r	38	36	p	53	41		68	64	e
9	5	p	24	0	s	39	10	c	54	22	p	69	3	e
10	0	i	25	14		40	12	e	55	4	s	70	8	
11	0	c	26	7	c	41	0	d	56	5	h	71	37	e
12	0	k	27	12		42	5	i	57	28	w	72	70	p
13	2	d	28	0	o	43	29	p	58	5	m	73	39	k
14	5	a	29	18		44	2	t	59	0	a			
15	9	e	30	6	c	45	38	i	60	0	n			

このように固まりを作ったら、「前の固まりの番号」と「加えた 1 文字」を順に符号化する。つまり、「0p0e0t2r0_1i1e0r5p0i0c0k2d5a9e...」のような並びを符号化する。固まりの個数は最大 255 個だとすれば 8 ビットで表わすことができるので、1 つの固まりにつき「前の固まり番号 8 ビット」+「アルファベット 1 文字 8 ビット」で 16 ビットになる。元の文章は 183 文字あったが、それが 73 個の固まりになったので結局 $73 \times (8+8) = 1168$ ビットとなる。

単純な符号化では、1 文字を 8 ビットで符号化するので $183 \times 8 = 1464$ ビット使う。これと比べると約 20% 短い表現になっていることが分かる。

Lempel-Ziv 方式では、文章を符号化する際に「以前見られた表現」を元にするので繰り返しの多い文章などは特に短くなる。例えば、上の詩と同じ内容がもう一度繰り返されていた場合、元の文章は 367 文字 (単純な符号化では 2936 ビット) になるが、「固まり」は 41 個増えるだけで計 114 個にしかならない。ビット数に換算すると 1824 ビットなので元の表現よりも 38% も短くなる。

3.3 損失の有無

ここまでで紹介した圧縮方法はいずれも、完全に元の情報が復元できるものであった。このような圧縮方法は**無損失の**(lossless) 符号化といわれる。

一方、画像や音声のようなアナログ情報は、ほぼ元に近い情報が復元できれば充分なこともある。このような場合には、画像や音声に特有の性質を利用して、**損失のある**(lossy) 圧縮方法も使われる。これについては画像や音声の表現形式の際に説明する。

4 暗号化

秘密にしたい内容を通信する場合には、盗聴される恐れのない手段を使って通信をするか、盗聴されても秘密が漏れないような通信内容にするかのどちらかである。一般に、盗聴の可能性がない通信手段を用意するのは容易ではないので、後者—**暗号化**が用いられる。

暗号化は、鍵 (パスワード) を使って、それを知らない限り通信内容が復元できないように符号化するものである。データを送る側と受ける側で同じ鍵を使うものは**秘密鍵**(または対称鍵) の、また、データを送る側と受ける側で違う鍵を使うものは**公開鍵**(または非対称鍵) の暗号化方式と呼ばれる。

4.1 秘密鍵暗号

秘密鍵暗号の中でも最も簡単なものは、アルファベットを 1 列に並べ、一定の文字数だけずらした位置のアルファベットに置き換えるものである。例えば次の図は 13 文字ずらす場合を示している:

```
abcdefghijklmnopqrstuvwxyz
    暗号化↓      ↑復号化
nopqrstuvwxyzabcdefghijklm
```

この場合、P は C に、e は r に置き換えらるので、Peter Piper の詩は次のようになる。

> crgre cvcre cvpxrq n crpx bs cvpxyrq crccref
> n crpx bs cvpxyrq crccref crgre cvcre cvpxrq
> vs crgre cvcre cvpxrq n crpx bs cvpxyrq crccref
> ubj znal cvpxyrq crccref qvq crgre cvcre cvpx

このとき「何文字ずらしたか」という情報を知らない限り、元の文章は復元できない。この「何文字ずらしたか」が鍵である。

この方法では鍵は 25 通りしかないので、鍵を知らない第 3 者であっても、全ての可能性を試みて元の文章を推測することができてしまう。「ずらす」かわりに、バラバラに置き換えることを考えたとすると 26 の階乗 つまり 25,165,824 通りの可能性があるので、全て試すことは難しいかも知れない。しかしそれでも、文字の出現頻度などの統計的な性質を使えば簡単に推測できてしまう。

実際の暗号化では、数文字分の情報をひとまとめにしてそれを置き換えることや、置き換えを何回も繰り返すことによって推測を防ぐようにしているため、推測をすることはできなくなっている。しかし、計算機の性能が高くなってきたため、膨大な数の鍵の可能性全てについて試してみることによって暗号を解読することは可能でなっている。

問題 7–7: (パスワードの解読) パスワードに使うことのできる文字は、アルファベットの大文字・小文字・数字の計 62 文字だとする。8 文字のパスワードが合っているかどうかを調べるのに、百万分の 1 秒 (10^{-6} 秒) かかるとする。8 文字のパスワードの全ての組み合わせを試すのに要する時間はどれだけか? もしパスワードにアルファベットの小文字だけしか使わない場合、この時間はどれだけになるか?

4.2 公開鍵暗号

公開鍵による暗号化は、暗号化する場合と、それを解読する場合で異なる鍵を使う。このような暗号化を使うと、秘密のメッセージを受け取りたい人は「自分だけしか知らない解読用の鍵」と「暗号化用の鍵」を作り、後者を送信者に渡し、それを使って暗号化したメッセージを受け取ればよい。このとき「暗号化用の鍵」では暗号化されたメッセージを解読できないので、「暗号化用の鍵」を知られても秘密が漏れることはない。

このような公開鍵による暗号化の 1 つに Rivest-Shamir-Adelman 法がある。この方法は (詳細は略すが)Fermat の定理を元にして暗号化と解読を行っている。また、暗号化用の鍵から解読用の鍵は、素因数分解によって求めることができるが、大きな数の素因数分解は非常に時間がかかることが知られているので実用上は推測できないものになっている。

公開鍵暗号は秘密鍵の受け渡しの必要がないため、色々な場面で使われている。とくに現在の計算機ネットワークシステムでは、電子メールメッセージの暗号化や、SSL (secure socket layer) を通した WWW のやりとり、電子署名などが代表的な使用例である。

5 実習: 情報発信

練習 7–8: (WWW ページの公開) HWB 「16.2 ホームページの開設」に従って WWW ページを作成し、公開せよ。

練習 7–9: (HTML の書き方) HWB 「16.3 HTML 文書の書き方」に従って以下のようなものを含む WWW ページを作成し公開せよ。

- 章や箇条書きなどの構造
- 知人や授業のページへのリンク

練習 7–10: (情報交換システムの比較) [課題の予告] 計算機を使った情報交換のためのシステムには、授業では紹介していないものも沢山ある。そのようなシステム 1 つをとりあげ、どのような仕組みで機能しているかを調べ、電子メールシステムあるいは WWW システムと比較せよ。

情報交換システムとしては、ネットニュース、チャット (インスタントメッセンジャー)、メリーリングリスト (メールマガジン)、掲示板 (WWW を使うもの・使わないもの)、オークション、電子商取引、ファイル交換 (Napster, WinMX, Winny, etc.)、weblog、IP 電話などが考えられる。

仕組みは図などを使って簡潔にわかりやすく説明せよ。(図については次回扱う。)

比較にあたっては「サーバどうしが通信をするか?」「利用者どうしは即座に情報を受け取るか?」「情報発信者の特定は可能か?」などの比較項目を考え、共通点や違いがわかるように整理せよ。

なお、引用する場合はレポート中のどこが引用範囲であるか、また、出典がどこであるかが明確になるようにせよ。断りなく他人の言説を書き写す行為は盗作になるので注意せよ。