符号理論･暗号理論

- No.5 ユニバーサル符号化 -

渡辺　裕

Coding Theory / Cryptography

- No.5 Universal Coding -

Hiroshi Watanabe

LZ77

- ユニバーサル, Ziv, Lempel (1977)
  - シンボルの出現確率に関する情報を事前に必要としない
  - データ数が多くなるほど圧縮効率が高くなる
- 可変長の符号列を固定長/可変長の符号に変換
  - VF (Variable to Fix) 符号
  - VV (Variable to Variable) 符号
- 辞書に基づいた符号化

LZ77

- Universal coding, Ziv, Lempel (1977)
  - Prior knowledge about symbol occurrence probability is not required
  - Compression efficiency is improved as the number of input data increases
- Variable length code is changed to Fixed/Variable length code
  - VF (Variable to Fix) code
  - VV (Variable to Variable) code
- Dictionary based coding

LZ77 (2)

- 符号化
  - 入力データに対して、参照バッファ(スライディングウインドウ)中に、符号化対象となるデータと同じパターンがあるかどうか探索
  - 同一パターンが存在すれば、[何データ前の位置][連続するデータ数][パターンの次のデータ]を固定長符号化

- 固定長符号化
  - スライディングウインドウ: データ8個->3bit
  - 連続数: 最大3bit
  - シンボルに必要なビット数(alphabet->5bit)

LZ77 (2)

- Encoding
  - Search the same pattern with the input data in the buffer (sliding window)
  - If the same pattern exists, [Back Position][Number of continuous data][next data] is fixed length coded

- Fixed length coding
  - Sliding window: If 8 data exists -> 3bit
  - Number of continuous data: ex. Max 3bit
  - Required bits for a symbol (ex. alphabet->5bit)

符号化例

- データ[AABCBBABCABCC...]の場合

位置	整合	次データ	出力
1		A	(0,0)A
2	A	B	(1,1)B
4		C	(0,0)C
5	B	B	(2,1)B
7	AB	C	(5,2)C
10	ABC	C	(3,3)C

- 圧縮比
  - (3+3+5)*6/13*5=66/65

Coding Example

- In case of data [AABCBBABCABCC…]

Location	Matching	Next data	Output
1		A	(0,0)A
2	A	B	(1,1)B
4		C	(0,0)C
5	B	B	(2,1)B
7	AB	C	(5,2)C
10	ABC	C	(3,3)C

- Compression Ratio
  - (3+3+5)*6/13*5=66/65

復号例

- 符号化データ(B,L)Cから位置とデータCを復号

入力	整合	次データ	出力
(0,0)A		A	A
(1,1)B	A	B	AAB
(0,0)C		C	AABC
(2,1)B	B	B	AABCBB
(5,2)C	AB	C	AABCBBABC
(3,3)C	ABC	C	AABCBBABCABCC

Decoding Example

- Recover location and data from Coded data (B,L)C

Input	Matching	Next data	Output
(0,0)A		A	A
(1,1)B	A	B	AAB
(0,0)C		C	AABC
(2,1)B	B	B	AABCBB
(5,2)C	AB	C	AABCBBABC
(3,3)C	ABC	C	AABCBBABCABCC

LZSS

- LZ77の改良版
  - 連続するデータ数Lがしきい値以上の場合
    - 位置Pを出力しL個移動
  - しきい値未満の場合
    - 参照バッファの最初のデータを出力し1個移動
- 冗長性の削減
  - (P,L)Cを常に符号化するのではなく、(P,L)あるいはCを符号化
- 実装例
  - PKZip, ARJ, LHArc, ZOO
  - Huffmanと組み合わされる

LZSS

- Improved version of LZ77
  - If the continuous data number is more than a threshold
    - Out put location P, move L data
  - Otherwise
    - Output the first data in the buffer and move one
- Reduce redundancy
  - Coding (P,L) or C selectively
- Implementation
  - PKZip, ARJ, LHArc, ZOO
  - Can be combined with Huffman coding

符号化例

- データ[AABBCBBAABC...]の場合 (L>=2に設定)

ステップ	位置	整合	出力
1	1		A
2	2	A	A
3	3		B
4	4	B	B
5	5		C
6	6	BB	(3,2)
7	8	AAB	(7,3)
8	11	C	C

Coding Example

- In case of [AABBCBBAABC...] (Set L>=2)

Step	Location	Matching	Output
1	1		A
2	2	A	A
3	3		B
4	4	B	B
5	5		C
6	6	BB	(3,2)
7	8	AAB	(7,3)
8	11	C	C

LZ78

- 動的辞書作成
  - 辞書にあるパターンとデータとの整合
  - 同じパターンなくなるまでデータを取り込む
  - 辞書の符号語とデータを出力

- 特長
  - 圧縮率はLZ77と同レベル
  - LZWがより一般的
  - LZ77に比べて比較演算量を削減

LZ78

- Dynamic creation of dictionary
  - Matching data with pattern in dictionary
  - Take data until the dame pattern disappears
  - Output words in dictionary and data

- Characteristics
  - Compression ration is the same level as LZ77
  - LZW is more popular
  - Matching complexity is lower than LZ77

符号化例

- データ[ABBCBCABA...]の場合

位置	辞書	次データ	出力
1	A (1)	A	(0,A)
2	B (2)	B	(0,B)
4	BC (3)	C	(2,C)
5	BCA (4)	A	(3,A)
8	BA (5)	A	(2,A)

Coding Example

- In case of data [ABBCBCABA...]

Location	Dictionary	Next data	Output
1	A (1)	A	(0,A)
2	B (2)	B	(0,B)
4	BC (3)	C	(2,C)
5	BCA (4)	A	(3,A)
8	BA (5)	A	(2,A)

LZW

- Lempel、Ziv、Welch
  - LZ78の改良
  - Deflate(LZH, ZIP)に比べて30%効率低下
  - DeflateではLZSS+Huffmanを採用
  - GIF, TIFFで使用

- 特許
  - Welchはスペリー社
  - スペリー+バロース->ユニシス(1986)
  - GIF特許問題->PNGの誕生

LZW

- Lempel、Ziv、Welch
  - Improved version of LZ78
  - Coding efficiency is 30% lower than Deflate(LZH, ZIP)
  - Deflate employs LZSS+Huffman
  - Used in GIF, TIFF
- Patent
  - Welch belonged Sperry Corporation
  - Sperry+Burroughs -> Unisis (1986)
  - GIF patent problem -> birth of PNG

LZW (2)

- 初期辞書が用意され、個々のデータがコード化されている
- 新規入力に対して辞書にあるパターン＋次データを新しく辞書に登録する
- 次データから再度パターン整合を行う

LZW (2)

- Initial dictionary is prepared, each data is preliminary coded
- Register a new pattern that consists of stored pattern in the dictionary and the next data
- Pattern matching is performed again to the next data

符号化例

- データ[ABBCBCABA...]の場合
  - 辞書にA=(1), B=(2), C=(3)が登録済み

位置	次データ	辞書登録	出力辞書
1	B	AB (4)	(1)
2	B	BB (5)	(2)
3	C	BC (6)	(2)
4	B	CB (7)	(3)
5	A	BCA (8)	(6)
7	A	ABA (9)	(4)

Coding Example

- In case of data [ABBCBCABA...]
  - A=(1), B=(2), C=(3) are registered in the dictionary

Position	Next date	Registration	Output
1	B	AB (4)	(1)
2	B	BB (5)	(2)
3	C	BC (6)	(2)
4	B	CB (7)	(3)
5	A	BCA (8)	(6)
7	A	ABA (9)	(4)

復号例

- 辞書にA=(1), B=(2), C=(3)が登録済み
- 復号側でも辞書を更新

ステップ	受信データ	復号データ	辞書登録
1	(1)	A	
2	(2)	B	AB (4)
3	(2)	B	BB (5)
4	(3)	C	BC (6)
5	(6)	BC	CB (7)
6	(4)	AB	BCA (8)

復号結果=ABBCBCAB...

Decoding Example

- A=(1), B=(2), C=(3) are registered in the dictionary
- Update the dictionary even at the decoder side

Step	Received data	Recovered Data	Registration
1	(1)	A	
2	(2)	B	AB (4)
3	(2)	B	BB (5)
4	(3)	C	BC (6)
5	(6)	BC	CB (7)
6	(4)	AB	BCA (8)

Decoded results = ABBCBCAB...