無失真壓縮演算法

PART II

教材出處

- 多媒體概論

  原著: Li Drew, 徐永煜 等譯 高立

  第七章 無失真壓縮演算法
- 資料壓縮

  戴顯權 著，紳藍。
- Introduction to Data Compression,

  Khalid Sayood, Morgan Kaufmann Publishers, Inc.,

目錄

PART I

- 緒論
- 消息理論基礎知識
- 變動碼長編碼
- 霍夫曼編碼
- 霍夫曼編碼的應用

PART II

- 以字典為本的編碼法
- 算術編碼
- 無失真JPEG
- 延伸閱讀

以字典為本的編碼法

- Dictionary-Based Compression
- This family of algorithm does not encode single symbols as bit streams, instead they encode phrases of variable length as single tokens.
- LZ77 (1977 by Ziv and Lempel) compression uses previously seen text as a dictionary.
- LZW by Terry Welch in 1984 - patent of USA.

以字典為本的編碼法

- Example :We use the Random House Dictionary of the English Language, 2nd Ed, Unabridged.

‘A good example of how dictionary based compression….’

1/1 822/3 674/4 1343/60 928/75 550/32 173/46 421/2

Page number

Item number in page

以字典為本的編碼法

- LZW演算法提供了定長度的字碼來表示像英文單字這樣具有不同長度的字元/符元的字串
- LZW的編解碼器在兩端會根據所收到的資料動態建立相同的字典
- LZW的過程是藉由把越來越長的重複字串放入字典中，而且如果有字串已經存在字典中，那麼 LZW 就會送出該字串編碼後的結果而不會送出字串本身

LZW 壓縮演算法

```
BEGIN
   s = next input character;
   while not EOF
       { c = next input character;

         if s + c exists in the dictionary
            s = s + c;
         else
           { output the code for s;
             add string s + c to the dictionary with a new code;
             s = c;
           }
        }
   output the code for s;
END
```

範例7.2 用LZW來壓縮字串ABABBABCABABBA

- 這裡我們用一個非常簡單的字典——有時也會把字典稱作字串表(string table)，一開始只有三個字元，而其所對應的編碼如下：

code	string
1	A
2	B
3	C

- 假如現在的輸入是ABABBABCABABBA，則LZW壓縮演算法的流程如下所述：

範例7.2 用LZW來壓縮字串 ABABBABCABABBA

s	c	output	code	string
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	A	4	10	ABA
A	B			
AB	B			
ABB	A	6	11	ABBA
A	EOF	1		

結果輸出的字碼就是1 2 4 5 2 3 4 6 1。所以總共輸出 9 個字碼而不是 14 個字元

LZW 解壓縮演算法

```
BEGIN
    s = NIL;
    while not EOF
        {
          k = next input code;
          entry = dictionary entry for k;
          output entry;
          if (s != NIL)
             add string s + entry[0] to dictionary with a new code;
          s = entry;
        }
END
```

範例7.3：對字串ABABBABCABABBA作 LZW 的解壓縮，輸入到解碼器的編碼是 1 2 4 5 2 3 4 6 1，一開始字串表的初始值是和編碼器的完全相同

LZW解壓縮演算法的流程如下述：

s	k	entry/output	code	string
			1	A
			2	B
			3	C
NIL	1	A		
A	2	B	4	AB
B	4	AB	5	BA
AB	5	BA	6	ABB
BA	2	B	7	BAB
B	3	C	8	BC
C	4	AB	9	CA
AB	6	ABB	10	ABA
ABB	1	A	11	ABBA
A	EOF			

很明顯可以看到輸出的結果就是ABABBABCABABBA，而這就說明了這的確是無失真壓縮。

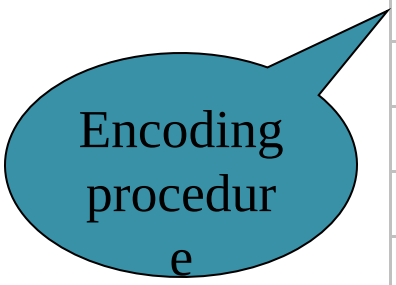
LZW

Example: I

Input string: "*wabba^wabba^wabba^wabba^woo^woo*"

Initial dictionary

code	phrase
1	^
2	a
3	b
4	o
5	w



input	output	dictionary	
symbol	code	code	phrase
w			
a	5	6	*wa*
b	2	7	*ab*
b	3	8	*bb*
a	3	9	*ba*
^	2	10	*a^*
w	1	11	*^w*
ab	6	12	*wab*
ba	8	13	*bba*
^w	10	14	*a^w*
abb	12	15	*wabb*

LZW

- Example 2
- Input string:“ WED WE WEE WEB WET...”

Initial dictionary code from 0-255

Characters Input	Code Output	Dictionary
“W”	‘^‘	256=“^W”
“E “	‘W’	257=“WE”
“D“	‘E’	258=“ED”
“^“	‘D’	259=“D^”
“WE”	256	260=“^WE”
“^“	‘E’	261=“E^“
“WEE”	260	262=“^WEE”
“^W”	261	263=“E^W”
“EB”	257	264=“WEB”
“^“	‘B‘	265=“B^“
“WET”	260	266=“^WET”
“^”	‘T‘	267=“T^”

LZW

LZW Decoding algorithm：

Initial:讀入第一個編號;
從字典中找出該編號所指之片語並輸出;
Step1: while not EOF
(讀入下一個編號;
從字典中找出該編號所指之片語並輸出;
將前一個解碼的片語加上現在解碼出的片語之第一個符號連起來，視為一新片語加入字典中;
)

LZW

Example 2:
Input codes : “ WED<256>E<260><261><257>B<260>T”

Input NEW-CODE	OLD-CODE	output string	character	New table entry
‘ ‘		“^“		
‘W’	‘ ‘	“W”	‘W’	256=“^W”
‘E’	‘W’	“E”	‘E’	257=“WE”
‘D’	‘E’	“D”	‘D’	258=“ED”
256	‘D’	“^W”	‘ ‘	259=“D^”
‘E’	256	“E”	‘E’	260=”^WE”
260	‘E’	“^WE”	‘ ‘	261=“E^“
261	260	“E^“	‘E’	262=“^WEE”
257	261	“WE”	‘W’	263=“E^W”
‘B’	257	“B”	‘B’	264=“WEB”
260	‘B’	“^WE”	‘ ‘	265=“B^“
‘T’	260	“T”	‘T’	266=”^WET”

Applications
File compression—UNIX compress

- The UNIX compress command is one of the earlier applications of LZW.
- The size of the dictionary is adaptive.
  - Start with the size of 512, using 9 bits for the codeword.
  - If filled up, the size is 1024, using 10 bits for the codeword.
  - Double the size until 2^{16}, then becomes a static dictionary.
- If the compression ratio falls below a threshold, then the dictionary building process is restarted.

Applications
Image Compression– The Graphic Interchange Format (GIF)

- GIF works well with computer-generated graphical images.
- It is another implementation of the LZW algorithm and is very similar to the compress command.
  - If a pixel using *b* bits, then the initial size of the dictionary is 2^{b+1}.
  - When the dictionary is filled up, the dictionary size is doubled until 4096.

Applications

Image Compression– The Graphic Interchange Format (GIF)

Sena

Sensin

Earth

Omaha

Comparison of GIF with arithmetic coding

image	GIF	Arithmetic Coding
Sena	51,085	53,431
Sensin	60,649	58,306
Earth	34,276	38,248
Omaha	61,580	56,061

算術編碼

- 算術編碼是比較晚近的一種編碼方法，而且在實作上常得到比霍夫曼編碼還好的效果
- 霍夫曼編碼是給每一個符元一個整數位元長度的字碼，對算術編碼來說，它是把全部的信息看成一個整體
- 一份信息(message)可以用一個半開區間[a,b)來表示，而a是介於0到1之間的實數。在一開始的時候區間會初始為[0,1)。隨著信息越來越長，區間長度也會越來越短，而其用於表達此區間的位元數也會隨之增加

演算法 7.5 算術編碼法的編碼器

```
BEGIN
   low = 0.0;   high = 1.0;   range = 1.0;

   while (symbol != terminator)
       {
         get (symbol);
         low =  low + range * Range_low(symbol);
         high = low + range * Range_high(symbol);
         range = high - low;
       }

   output a code so that low <= code < high;
END
```

範例：算術編碼

Symbol	Probability	Range
A	0.2	[0, 0.2)
B	0.1	[0.2, 0.3)
C	0.2	[0.3, 0.5)
D	0.05	[0.5, 0.55)
E	0.3	[0.55, 0.85)
F	0.05	[0.85, 0.9)
$	0.1	[0.9, 1.0)

(a)符元出現的機率分布

圖7.8 算術編碼：對 CAEE$ 作編碼

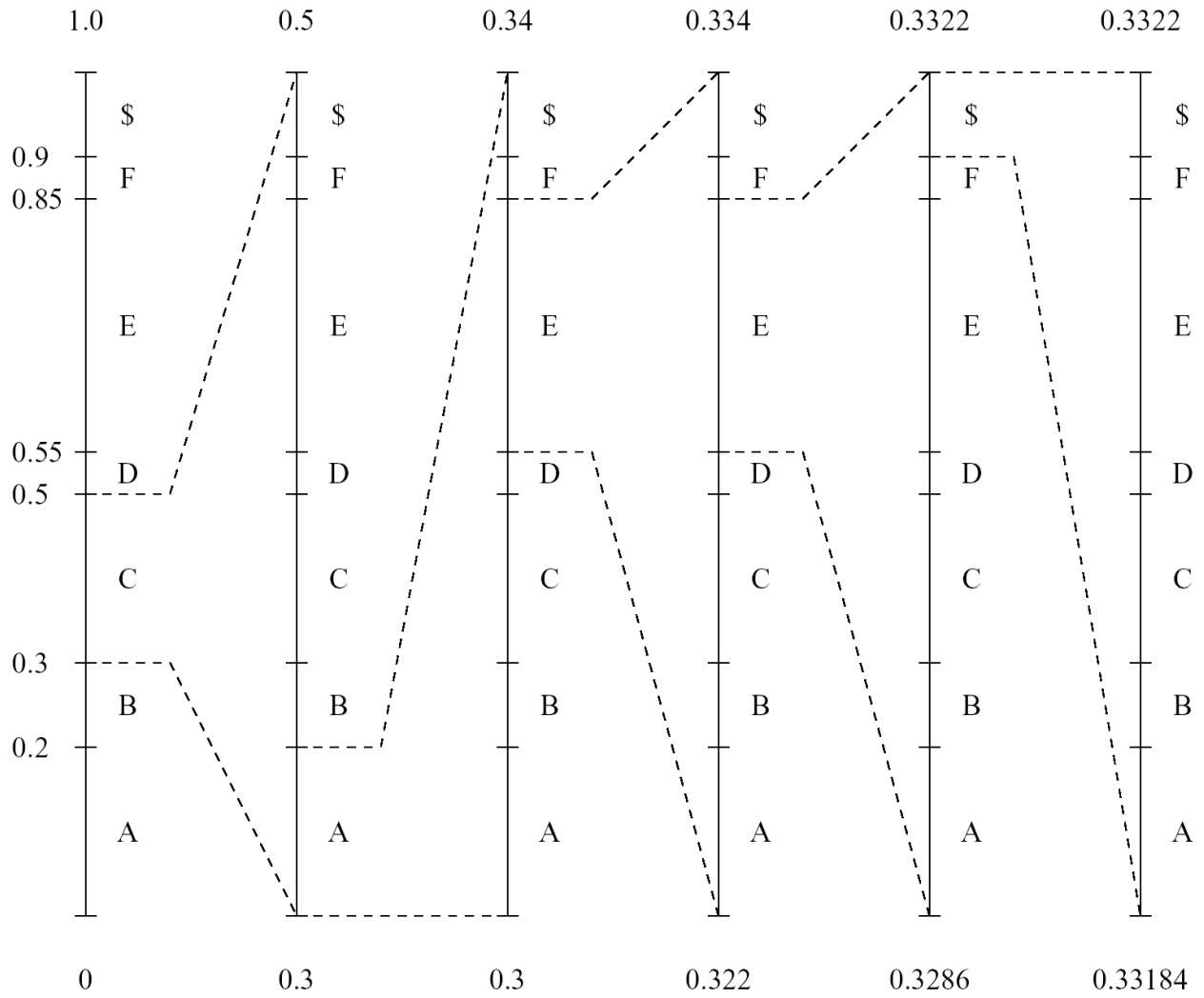


圖7.8(b)對 CAEE$ 區間減縮的示意圖

Symbol	low	high	range
	0	1.0	1.0
C	0.3	0.5	0.2
A	0.30	0.34	0.04
E	0.322	0.334	0.012
E	0.3286	0.3322	0.0036
$	0.33184	0.33220	0.00036

(c) 產出的新區間和其對應的高低值

圖7.8（續） 算術編碼：對 CAEE$ 作編碼

流程 7.2 編碼器用的字碼產生流程

```
BEGIN
    code = 0;
    k = 1;
    while (value(code) < low)
        { assign 1 to the kth binary fraction bit
          if (value(code) > high)
             replace the kth bit by 0

          k = k + 1;
        }
END
```

在編碼的最後一步是要找出一個落在[*low*，*high*)區間的數值，上面的演算法可以確保在*low*和*high*的區間內找到最短的二進位表示的字碼

演算法7.6 算術編碼的解碼器

```
BEGIN
   get binary code and convert to
       decimal value = value(code);
   Do
       { find a symbol s so that
              Range_low(s) <= value < Range_high(s);
         output s;
         low = Rang_low(s);
         high = Range_high(s);
         range = high - low;
         value = [value - low] / range;
       }
   Until symbol s is a terminator
END
```

表7.5 算術編碼法：符元 CAEE$ 的解碼

value	Output Symbol	low	high	range
0.33203125	C	0.3	0.5	0.2
0.16015625	A	0.0	0.2	0.2
0.80078125	E	0.55	0.85	0.3
0.8359375	E	0.55	0.85	0.3
0.953125	$	0.9	1.0	0.1

無失真JPEG

- **無失真JPEG**：JPEG影像壓縮中的特例
- 具有預測性的方法
  1. **建立差動預測**：預測器會拼湊目前被預測點X鄰近三個像素的值當作現在被預測點的值，就像在圖7.10所示，而預測器可以使用在表7.6所列的七種方式其中之一
  2. **編碼**：編碼器會比較預測值和X的實際值，根據之間的差值用任何我們曾提過的無失真壓縮技術，像是霍夫曼方式來作編碼

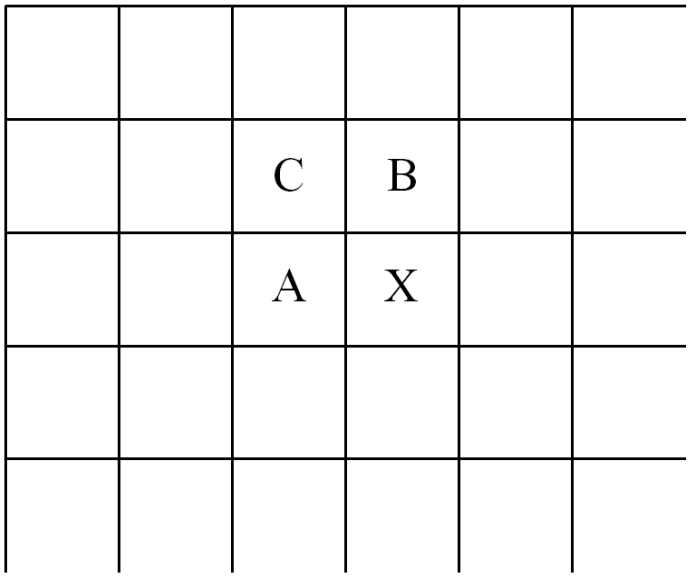


圖7.10 在無失真JPEG中，被預測者的相鄰像素

- **注意**：於編碼/解碼的流程中，在解碼端若欲預測某點，則之前必須注意到對於該點的A,B,C這些位置上的點都必須是已經解碼出來的

表7.6 無失真 JPEG 的預測器

Predictor	Prediction
P1	A
P2	B
P3	C
P4	A + B - C
P5	A + (B - C) / 2
P6	B + (A - C) / 2
P7	(A + B) / 2

表7.7 JPEG非失真壓縮與其他非失真壓縮程式的比較

Compression Program	Compression Ratio			
	Lena	football	F-18	flowers
Lossless JPEG	1.45	1.54	2.29	1.26
Optimal lossless JPEG	1.49	1.67	2.71	1.33
compress (LZW)	0.86	1.24	2.21	0.87
gzip (LZ77)	1.08	1.36	3.10	1.05
gzip -9 (optimal LZ77)	1.08	1.36	3.13	1.05
pack (Huffman coding)	1.02	1.12	1.19	1.00

延伸閱讀

- **書面資料：**
  - *The Data Compression Book* by M. Nelson
  - *Introduction to Data Compression* by K. Sayood
- **網站：**到第七章之延申閱讀的連結，其中包括
  - 由尼爾森(Mark Nelson)所編集的相當豐富的資源
  - 介紹資料壓縮基礎理論的網頁
  - comp.compression 和 comp.compression.research 新聞群組的常見問題集(FAQ)
  - 一組非失真壓縮法的次應用程式(applets)
  - 關於算數編碼的簡介
  - 灰階的測試影像f-18.bmp，flowers.bmp，football.bmp，lena.bmp