KOD ARYTMETYCZNY

KODOWANIE DANYCH, A.Przelaskowski

- Pomysł jednej liczby
- Koncepcja zwężanego przedziału
- Algorytmy kodu
- Implementacje całkowitoliczbowe

Pomysł jednej liczby

- Zamiast relacji prawdopodobieństwo $\rightarrow$ słowo kodowe pozostajemy jedynie z określonym prawdopodobieństwem
- Właściwy opis prawdopodobieństw (kumulacja):

  $A_S=\{a_1,\ldots,a_n\}$ i $P_S=\{p_1,\ldots,p_n\} \rightarrow F_S=\{p_1,p_1+p_2,\ldots,p_1+\ldots+p_{n-1}\}$

  czyli $F_S=\{F(a_i): i= 1,\ldots,n-1\}$, $F(a_i)=\sum_{j=1}^{i} p_j$

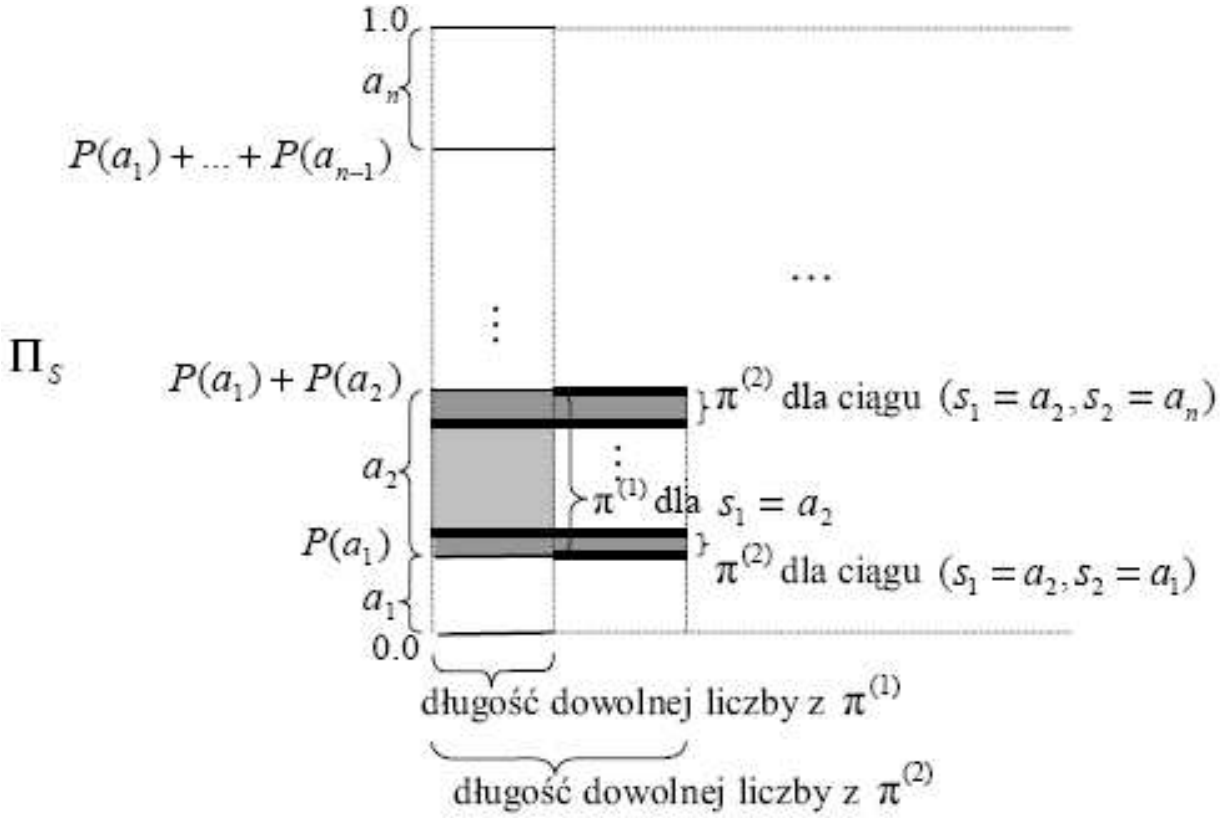
- Przyporządkowanie przedziału:

$$\pi_s : \ a_i \in A_S \rightarrow [F(a_{i-1}), F(a_i)) \subset [0,1)$$

- Ogólniej:

$$\pi_{[d,g)} : \ a_i \in A_S \rightarrow [d + F(a_{i-1}) \cdot (g-d), d + F(a_i) \cdot (g-d)) \subset [d,g)$$

Modyfikacja przedziału (zasadnicza koncepcja)



kodowanie to modyfikacja przedziału kodu $\pi^{(i)}= \pi(a_1,a_2,\ldots a_i)= \pi(a_i, \pi^{(i-1)})$

zasada zawężania przedziału: $\pi^{(i)} \subset \pi^{(i-1)}$

Kodowanie i dekodowanie

- Koder: $\pi^{(i)} = \left[D^{(i)}, G^{(i)}\right), \; i = 1,2,\ldots$

$$D^{(i)} = D^{(i-1)} + \left(G^{(i-1)} - D^{(i-1)}\right) F(a_{k-1})$$

$$G^{(i)} = D^{(i-1)} + \left(G^{(i-1)} - D^{(i-1)}\right) F(a_k)$$

liczba kodowa: dowolna $\mathcal{L}^t \in \pi^{(t)}$

- Dekoder (dwa sposoby):
  - przeskalowanie liczby kodowej do linii prawdopodobieństw

$$\mathcal{L}^t = \mathcal{L}^{(1)}, \qquad \mathcal{L}^{(i+1)} = \frac{\mathcal{L}^{(i)} - F(a_{k-1})}{F(a_k) - F(a_{k-1})}$$

normalizacja $\pi^{(1)}$

  - śledzenie przedziału kodowego z ograniczoną dokładnością liczby kodowej

$$s_i = a_k \Leftrightarrow \frac{\left[\mathcal{L}^t\right] - D^{(i)}}{G^{(i)} - D^{(i)}} \in \left[F(a_{k-1}), F(a_k)\right]$$

normalizacja $\pi^{(i)}$

Przykład - kodowanie

a_i	$P(a_i)$	$\pi_s(a_i)$
„A”	2/10	[0,2/10)
„E”	1/10	[2/10,3/10)
„K”	1/10	[3/10,4/10)
„M”	1/10	[4/10,5/10)
„R”	1/10	[5/10,6/10)
„T”	2/10	[6/10,8/10)
„Y”	2/10	[8/10,1)

Wejście = „ARYTMETYKA"

linia prawdopodobieństw

i	Kolejne symbole wejściowe $s_i = a_k$	$D^{(i)}$	$G^{(i)}$	$R^{(i)}$
0	inicjalizacja	0	1	1
1	„A”	0,0	0,2	0,2
2	„R”	0,1	0,12	0,02
3	„Y”	0,116	0,12	0,004
4	„T”	0,1184	0,1192	0,0008
5	„M”	0,11872	0,1188	0,00008
6	„E”	0,118736	0,118744	0,000008
7	„T”	0,1187408	0,1187424	0,0000016
8	„Y”	0,11874208	0,1187424	0,00000032
9	„K”	0,118742176	0,118742208	0,000000032
10	„A”	0,118742176	0,1187421824	0,0000000064

Przykład - dekodowanie

a_i	$P(a_i)$	$\pi_s(a_i)$
„A”	2/10	[0,2/10)
„E”	1/10	[2/10,3/10)
„K”	1/10	[3/10,4/10)
„M”	1/10	[4/10,5/10)
„R”	1/10	[5/10,6/10)
„T”	2/10	[6/10,8/10)
„Y”	2/10	[8/10,1)

i	Liczba kodowa $\mathcal{L}^{(i)}$	Symbole dekodowane $s_i = a_k$	$F(a_{k-1})$	$F(a_k)$	$F(a_k) - F(a_{k-1})$
1	0,118742176	„A”	0	0,2	0,2
2	0,59371088	„R”	0,5	0,6	0,1
3	0,9371088	„Y”	0,8	1	0,2
4	0,685544	„T”	0,6	0,8	0,2
5	0,42772	„M”	0,4	0,5	0,1
6	0,2772	„E”	0,2	0,3	0,1
7	0,772	„T”	0,6	0,8	0,2
8	0,86	„Y”	0,8	1	0,2
9	0,3	„K”	0,3	0,4	0,1
10	0	„A”	0	0,2	0,2
...	0	...	...	...	...

Implementacja całkowitoliczbowa

- rejestry o ograniczonej długości, np. 16 bitowe
- przedział niedomknięty z góry [0, $0{,}1111_2$….)
- usuwamy 0, i uzupełniamy DOL zerami i GORA
- mamy więc

  DOL=0x0000 oraz GORA=0xFFFF

  szerokość przedziału: GORA-DOL+1
- wysuwanie najbardziej znaczącej cyfry, gdy jest jednakowa w DOL i GORA i uzupełnianie zerami (DOL) oraz jedynkami (GORA)
- eliminacja niedomiaru (najstarsza cyfra różna o 1, kolejna to 0 w GORA i 9 (1_2) w DOL:
  - najbardziej znaczące cyfry pozostają bez zmian, zaś pozostałe przesuwane są o jedną pozycję w lewo (giną niewygodne 0 i 9 na pozycji niedomiaru)
  - najmłodsza pozycja w rejestrach zostaje odpowiednio uzupełniona
  - inkrementacja licznika niedomiarów

Przykład: kodowanie całkowitoliczbowe

Kolejne s_i i działania pomocnicze	DOL	GORA	GORA-DOL+1	Ciąg kodowy
Inicjalizacja	00000	99999	100000	–
„A”	00000	19999	020000	
„R”	10000	11999		
Wysuń 1	00000	19999	020000	1
„Y”	16000	19999		
Wysuń 1	60000	99999	040000	11
„T”	84000	91999	008000	
„M”	87200	87999		
Wysuń 8 i 7	20000	99999	080000	1187
„E”	36000	43999	008000	
„T”	40800	42399		
Wysuń 4	08000	23999	016000	11874
„Y”	20800	23999		
Wysuń 2	08000	39999	032000	118742
„K”	17600	20799	003200	
„A”	17600	18239		
Wysuń 1	76000	82409		1187421
Wysuń 7 i 6				118742176

Przykład: dekodowanie całkowitoliczbowe

KOD	DOL	GORA	GORA-DOL+1	$\mathcal{L}_{\text{lok}}$	Dekodowane s_i i działania pomocnicze
11874	00000	99999	100000	0,11874	„A”
	00000	19999	020000	0,5937	„R”
	10000	11999			Usuń 1
18742	00000	19999	020000	0,9371	„Y”
	16000	19999			Usuń 1
87421	60000	99999	040000	0,685525	„T”
	84000	91999	008000	0,427625	„M”
	87200	87999			Usuń 8 i 7
42176	20000	99999	080000	0,2772	„E”
	36000	43999	008000	0,772	„T”
	40800	42399			Usuń 4
21760	08000	23999	016000	0,86	„Y”
	20800	23999			Usuń 2
17600	08000	39999	032000	0,3	„K”
	17600	20799	003200	0	„A”

Algorytmy kodera (alfabet binarny)

- szacowanie statystyki
- inicjalizacja: DOL=0…..0, GORA=1…..1 oraz licznik niedomiaru $L = 0$
- modyfikacje: PRZE = GORA − DOL + 1

$$\text{DOL} = \text{DOL} + \text{PRZE} \cdot F(a_k - 1)$$

$$\text{GORA} = \text{DOL} + \text{PRZE} \cdot F(a_k)$$

- jeśli najstarszy bit b w DOL i GORA jest jednakowy:
  - cyfry obu rejestrów przesuń o jedną pozycję w lewo, a wysuniętą najstarszą cyfrę prześlij na wyjście
  - uzupełnij DOL zerem oraz GORA jedynką
  - jeśli licznik $L > 0$, dosyłanych jest L bitów $(1 - b)$; $L = 0$
- jeśli na najstarszych bitach w GORA i DOL jest odpowiednio 10_2 i 01_2, to przesuń w lewo bity obu rejestrów z wyjątkiem dwu najbardziej znaczących i uzupełnij rejestry; $L = L + 1$
- zakończenie: dopisz do wyjścia wszystkie znaczące bity z DOL

Algorytm dekodera

- Pobierz linię prawdopodobieństw
- Ustal DOL=0….0, GORA=1....1 oraz licznika *i* = 1
- Wczytaj do KOD η najstarszych bitów liczby kodowej
- Oblicz lokalną liczbę kodową:

$$\mathcal{L}_{\text{lok}} = \frac{\texttt{KOD} - \texttt{DOL}}{\texttt{GORA} - \texttt{DOL} + 1}$$

- Dekoduj symbol:

$$s_i = a_k \Leftrightarrow \mathcal{L}_{\text{lok}} \in \left[F(a_{k-1}), F(a_k)\right)$$

- Modyfikuj DOL i GORA jak w koderze: KOD uzupełnij kolejnymi bitami liczby kodowej, w sytuacji niedomiaru postępuj analogicznie jak w koderze
- Licz dekodowane symbole *i* =*i*+1

Procedury kodera (implem. C)

```
/* Inicjalizacja globalnych zmiennych kodowania */

unsigned short int dol;
unsigned short int gora;
long niedomiar;

...
...

void inicjalizuj_koder()
{
 dol=0;
 gora=0xffff;
 niedomiar=0;
}

...
...
```

```
/* Funkcja kodowania symbolu s */
void koduj_symbol(ZBIOR_BIT *zbior, SYMBOL *s, int
                  s_dol, int s_gora, int calk_symbol)
/* do zbior zapisywany jest ciąg kodowy, s_dol i s_gora to
przedział skumulowanych częstości występowania s przy danym
kontekście, a calk_symbol to całkowita liczba wystąpień symb
przy danym kontekście (przy zerowym rzędzie modelu jest
to liczba wszystkich zakodowanych dotąd symboli */
{

 long przedzial;

 /* Aktualizacja przedziału kodowego */
 przedzial=(long)(gora-dol)+1;
 gora=dol+(unsigned short int)((przedzial*s_gora)/calk_symbol-1);
 dol=dol+(unsigned short int)((przedzial*s_dol)/calk_symbol);
```

```
/* Normalizacja zmiennych przedziału */
for(;;) {
 if((gora & 0x8000)==(dol & 0x8000)) {
                                /* Najstarszy bit jest
                                   jednakowy */
  WyslijBit(zbior,gora & 0x8000);
                                /* Zapisanie najstarszego bitu
                                   obu rejestrów */
  while(niedomiar>0) {
   WyslijBit(zbior,~gora & 0x8000);
                                /* Zapisanie negacji najstarszego
                                   bitu obu rejestrów */
   niedomiar--;
  }
 } else if((dol & 0x4000) && !(gora & 0x4000)) {
                                /* Wykrycie zagrożenia
                                   niedomiarem */
  niedomiar+=1;
  dol&=0x3fff;
  gora|=0x4000;
 } else
  return;

 dol<<=1;
 gora<<=1;
 gora|=1;
}
}
```

Procedury dekodera (implem. C)

```
/* Inicjalizacja globalnych zmiennych dekodowania */

unsigned short int kod; /* Zmienna, do której wpisywana
                           jest reprezentacja kodowa */
unsigned short int dol;
unsigned short int gora;

...
...

void inicjalizuj_dekoder(ZBIOR_BIT *zbior )
/* zbior to zbior bitowy zawierający ciąg danych kodowych
{
 kod=0;
 for(i=0;i<16;i++) { /* Wczytanie pierwszych 16 bitów
   kod<<=1;
   kod+=PobierzBit(zbior);
  }
 dol=0;
 gora=0xffff;
}

...
...
```

```
/* Funkcja dekodowania symbolu s */
int dekoduj_symbol(ZBIOR_BIT *zbior, short int kum_waga[])
/* ze zbior czytany jest ciąg kodowy; kum_waga to tablica
skumulowanych częstości wystąpień kolejnych symboli alfabetu
w porządku malejącym, tj. kum_waga[0] zawiera liczbę wszystkich
wystąpień danego kontekstu */
{
 long przedzial;
 short int calk_symbol;
 int s;

 calk_symbol)=kum_waga[0];
 przedzial=(long)(gora-dol)+1;
 liczba_kod=
   (short int)((((long)(kod-dol)+1)*calk_symbol-1)/przedzial);
 for(s=0;liczba_kod<kum_waga[s];s++);
                                  /* Dekodowanie symbolu poprzez
                                     rzutowanie liczby kodowej na
                                     linię prawdopodobieństw */
 s_gora=kum_waga[s-1];            /* Ustalenie podprzedziału
                                     symbolu s*/
 s_dol=kum_waga[s];
 /* Aktualizacja przedziału kodowego */
 przedzial=(long)(gora-dol)+1;
 gora=dol+(unsigned short int)((przedzial*s_gora)/calk_symbol-1);
 dol=dol+(unsigned short int)((przedzial*s_dol)/calk_symbol);

 /* Normalizacja zmiennych przedziału */
 for(;;) {
  if((gora & 0x8000)==(dol & 0x8000)) {
  } else if((dol & 0x4000)==0x4000 && (gora & 0x4000)==0) {
   kod ^= 0x4000;
   dol &= 0x3fff;
   gora |= 0x4000;
  } else
   return(s);
  dol<<=1;
  gora<<=1;
  gora|=1;
  kod<<=1;
  kod+=PobierzBit(zbior); /* Wczytanie kolejnego bitu ciągu kodowego
 }
```