# Podstawowe pojęcia. Teoria informacji

Kodowanie i kompresja danych - Wykład 1

Maciek Gębala

26 luty 2020



Notatki

## Literatura


- D. Salomon, G. Motta, *Handbook of Data Compression*, Springer-Verlag London 2010 (ISBN: 978-1-84882-903-9)
- K. Sayood, *Kompresja danych - wprowadzenie*, READ ME 2002 (ISBN 83-7243-094-2)
- J. Adamek, *Fundations of Coding*, Wiley 1991 (ISBN 0-47-162187-0)
- R. Hamming, *Coding and Information Theory*, Prentice-Hall (ISBN 0-13-139139-1)
- A. Przelaskowski, *Kompresja danych*, BTC 2005 (ISBN: 83-60233-05-5)




Notatki

## Rodzaje kodowania

**Kodowanie**

Przyporządkowanie elementom jakiegoś alfabetu ciągów binarnych (lub ciągów nad innym alfabetem).

Kodowanie może mieć różne cele:

- Zmniejszenie objętości danych – kompresja.
- Zapewnienie odporności na błędy – kody korekcyjne.
- Zapewnienie poufności danych – kryptografia.



Notatki

## Kompresja bezstratna i stratna

**Kompresje bezstratna (lossless compression)**

Z postaci skompresowanej można (zawsze!) odtworzyć postać danych identyczną z oryginałem.

**Kompresja stratna (lossy compression)**

Algorytm dopuszcza pewien poziom utraty informacji w zamian za lepszy współczynnik kompresji. Uwaga: W niektórych zastosowaniach może być to niebezpieczne! (np. obrazy medyczne)



Notatki

## Rodzaje kodów

Kody stałej długości – np. kody ASCII o długości 8 bitów. (Ponieważ długość jest stała nie ma kłopotu z podziałem na znaki.)

Kody o różnej długości – kody prefiksowe, kod Morse'a. (Ważne jest zapewnienie, że kody da się prawidłowo odczytać.)



Notatki

## Kody jednoznaczne

| Litera | Prawd. | kod 1 | kod 2 | kod 3 | kod 4 |
|---|---|---|---|---|---|
| $a_1$ | 0,5 | 0 | 0 | 0 | 0 |
| $a_2$ | 0,25 | 0 | 1 | 10 | 01 |
| $a_3$ | 0,125 | 1 | 00 | 110 | 011 |
| $a_4$ | 0,125 | 10 | 11 | 111 | 111 |
| średnia długość | | 1,125 | 1,25 | 1,75 | 1,75 |

- kod 2: 1111 - czego to jest kod?
- kod 3: kod prefiksowy - żaden kod nie jest prefiksem drugiego.
- kod 4: 011111111111: co jest pierwszą literą?



Notatki

## Techniki kompresji

- Dwa algorytmy - kompresja i dekompresja.
- Kompresja bezstratna – dane po kompresji i dekompresji są identyczne z danymi wejściowymi.
- Kompresja stratna – dane wejściowe są wstępnie przetwarzane aby zwiększyć stopień kompresji ale po dekompresji są różne od wejściowych.

### Miary jakości kompresji

- Stosunek liczby bitów danych do bitów po kompresji.
- Procentowy stosunek bitów po kompresji do bitów danych.
- Czas kompresji i dekompresji – w zależności od sposobu użycia może być ważny.



Notatki

## Modelowanie danych

### Przykład 1

- Weźmy ciąg:
  *9 11 11 11 14 13 15 17 16 17 20 21*
- Do zapamiętania tego ciągu dosłownie potrzebujemy 5 bitów na każdą liczbę.
- Weźmy wzory $\hat{x}_n = n + 8$ i $e_n = x_n - \hat{x}_n$.
- Wówczas ciąg $e_n$ ma postać:
  *0 1 0 -1 1 -1 0 1 -1 -1 1 1*
- Teraz dla każdego elementu nowego ciągu wystarczą 2 bity.
- Dane spełniają w przybliżeniu pewną regułę.



Notatki

## Modelowanie danych

**Przykład 2**

- Weźmy ciąg:
  *27 28 29 28 26 27 29 28 30 32 34 36 38*
- Do zapamiętania tego ciągu dosłownie potrzebujemy 6 bitów na każdą liczbę.
- Każda wartość w ciągu jest bliska poprzedniej.
- Weźmy wzory $e_n = x_n - x_{n-1}$, $e_1 = x_1$. Nowy ciąg ma postać:
  *27 1 1 -1 -2 1 2 -1 2 2 2 2 2*
- Teraz ciąg do zakodowania jest prostszy.



Notatki

## Modelowanie danych

**Przykład 3**

- Weźmy tekst:
  *a_barayaran_array_ran_far_faar_faaar_away*
- Mamy 8 różnych symboli, więc na każdy potrzebujemy 3 bity.
- Jeśli użyjemy kodu z tabelki poniżej, to zostanie użytych 106 bitów zamiast 123 (2,58 bitu na symbol).

| a | _ | b | f | n | r | w | y |
|---|---|---|---|---|---|---|---|
| 1 | 001 | 01100 | 0100 | 0111 | 000 | 01101 | 0101 |

- Wykorzystujemy własności statystyczne danych.



Notatki

## Zagadka

**Czy istnieje algorytm kompresji bezstratnej, który skompresuje wszystkie podane mu dane?**

- Kompresja bezstratna jest funkcją różnowartościową.
- Różnych danych długości $n$ jest $2^n$ (ciągi bitów).
- Różnych ciągów bitowych długości mniejszej niż $n$ jest $2^n - 1$. ($\sum_{i=1}^{n-1} 2^i$)
- **Wniosek 1:** istnieje ciąg który nie może być skompresowany.
- **Wniosek 2:** ponad połowa ciągów skróci się o co najwyżej 1 bit.



Notatki

## Teoria informacji

Jeśli $P(A)$ jest prawdopodobieństwem wystąpienia informacji $A$ to niech

$$i(A) = \log \frac{1}{P(A)} = -\log P(A)$$

będzie miarą tej informacji.

Jeśli $A$ i $B$ są niezależne, to

$$\begin{aligned} i(AB) &= \log \frac{1}{P(AB)} = \log \frac{1}{P(A)P(B)} = \\ &= \log \frac{1}{P(A)} + \log \frac{1}{P(B)} = i(A) + i(B) \end{aligned}$$

Podstawą logarytmu jest 2 a jednostką informacji *bit*.



Notatki

## Teoria informacji

Załóżmy, że mamy zbiór wiadomości $A_1, \ldots, A_n$, które pojawiają się z prawdopodobieństwami $P(A_1), \ldots, P(A_n)$ ($\sum_{i=1}^{n} P(A_i) = 1$).

Średnia informacja w tym zbiorze jest określona wzorem

$$H = \sum_{i=1}^{n} P(A_i) i(A_i)$$

Wielkość tą nazywamy *entropią*.

Kody jednoznacznie dekodowalne w modelu z niezależnymi wystąpieniami symboli muszą mieć średnią długość co najmniej równą entropii.



Notatki

## Przykład

- Weźmy ciąg *1 2 3 2 3 4 5 4 5 6 7 8 9 8 9 10*
- $P(1) = P(6) = P(7) = P(10) = \frac{1}{16}$, $P(2) = P(3) = P(4) = P(5) = P(8) = P(9) = \frac{2}{16}$
- $H = -\sum_{i=1}^{10} P(i) \log P(i) = 3,25$
- Najlepszy schemat kodujący ten ciąg wymaga 3,25 bitu na znak.
- Jeśli jednak założymy, że elementy ciągu nie są niezależne i zastąpimy ciąg różnicami to otrzymamy *1 1 1 -1 1 1 1 -1 1 1 1 1 -1 1 1*



Notatki

## Przykład

- Weźmy ciąg *1 2 1 2 3 3 3 3 1 2 3 3 3 3 1 2 3 3 1 2*
- Prawdopodobieństwa wynoszą: $P(1) = P(2) = \frac{1}{4}$ i $P(3) = \frac{1}{2}$.
- Entropia jest równa 1,5 bitu na znak.
- Jeśli jednak weźmiemy bloki złożone z dwóch znaków to $P(12) = \frac{1}{2}$ i $P(33) = \frac{1}{2}$, czyli entropia jest równa 1 bit na parę (0,5 bitu na znak).



Notatki

## Test na jednoznaczną dekodowalność

Kod $a$ jest prefiksem kodu $b$, jeśli $b$ jest postaci $ax$. $x$ nazywamy sufiksem $b$ względem $a$.

**Algorytm**

Tworzymy listę słów kodowych.
Dla każdej pary sprawdzamy, czy jedno słowo jest prefiksem drugiego. Jeśli tak to do listy dodajemy sufiks drugiego słowa (chyba, że już dodaliśmy taki sufiks).
Powtarzamy powyższą procedurę, aż do momentu kiedy znajdziemy na liście sufiks równy słowu kodowemu (*kod nie jest jednoznaczny*), albo nie można znaleźć nowych sufiksów (*kod jest jednoznaczny*).



Notatki

## Przykład

- Weźmy kod *{0, 01, 11}*.
- Kod *0* jest prefiksem *01*. Innych par nie ma, więc nowa lista ma postać *{0, 01, 11, 1}*.
- Teraz dla tej listy mamy *0* jako prefiks *01* i *1* jako prefiks *11*, ale sufiks *1* już dopisaliśmy do listy, więc lista się nie zmienia.
- Kod jest więc jednoznacznie dekodowalny.



Notatki

## Przykład

- Weźmy kod *{0, 01, 10}*.
- Kod *0* jest prefiksem *01*. Innych par nie ma, więc nowa lista ma postać *{0, 01, 10, 1}*.
- Teraz dla tej listy mamy *0* jako prefiks *01*, ale on już jest na liście, oraz *1* jako prefiks *10*, ale sufiks *0* jest równy kodowi *0*.
- Kod nie jest więc jednoznacznie dekodowalny.



Notatki

## Kody prefiksowe

Kod w którym żadne słowo kodowe nie jest prefiksem innego słowa kodowego. (Łatwo zauważyć, że kod prefiksowy jest jednoznacznie dekodowalny.)

**Nierówność Krafta-McMillana**

Niech $C$ będzie kodem składającym się z $N$ słów o długościach $l_1, l_2, \ldots, l_N$. Jeżeli $C$ jest jednoznacznie dekodowalny, to

$$K(C) = \sum_{i=1}^{N} \frac{1}{2^{l_i}} \leqslant 1.$$



Notatki

## Nierówność Krafta-McMillana - dowód

- Jeśli $a > 1$, to $a^n$ dąży do nieskończoności. Jeśli $a^n$ nie rośnie, to $a \leqslant 1$.
- Dla dowolnego $n$

$$[K(C)]^n = \left(\sum_{i=1}^{N} \frac{1}{2^{l_i}}\right) \cdot \ldots \cdot \left(\sum_{i=1}^{N} \frac{1}{2^{l_i}}\right) = \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} \cdots \sum_{i_n=1}^{N} \frac{1}{2^{l_{i_1}+l_{i_2}+\ldots+l_{i_n}}}.$$

- Wykładnik $l_{i_1} + l_{i_2} + \ldots + l_{i_n}$ jest sumą długości $n$ słów z kodu $C$. Niech $l = \max\{l_1, l_2, \ldots, l_N\}$. Wtedy maksymalny wykładnik jest równy $nl$.
- Powyższą sumę można zapisać jako

$$[K(c)]^n = \sum_{k=n}^{nl} A_k 2^{-k}$$

gdzie $A_k$ jest sumą kombinacji $n$ słów kodowych o długości $k$.



Notatki

## Nierówność Krafta-McMillana - dowód

- Istnieje $2^k$ różnych ciągów binarnych długości $k$. Ale jeśli kod jest jednoznacznie dekodowalny, to każdy taki ciąg może reprezentować tylko jeden ciąg słów kodowych. Czyli
$$A_k \leqslant 2^k$$
- Stąd
$$[K(C)]^n \leqslant \sum_{k=n}^{nl} 2^k 2^{-k} = nl - n + 1$$
- Ale jeśli $K(C)$ byłoby większe od 1, to $[K(C)]^n$ rosłoby wykładniczo, a według powyższej nierówności rośnie liniowo. Czyli
$$K(C) \leqslant 1.$$



Notatki

## Kody Shannon-Fano

- Niech symbole $a_i$ występują odpowiednio z prawdopodobieństwami $p_i$.
- Weźmy długości kodów $l_i = \lceil -\log p_i \rceil$.
- Długości $l_i$ spełniają nierówność Krafta-McMillana.
$$\sum_{i=1}^{N} \frac{1}{2^{l_i}} \leqslant \sum_{i=1}^{N} \frac{1}{2^{-\log p_i}} = \sum_{i=1}^{N} p_i = 1$$
- Istnieje więc kod prefiksowy o takich długościach.
- Łatwo zauważyć, że średnia długość tego kodu jest nie większa niż entropia plus 1.



Notatki

## Konstrukcja kodu o podanych długościach

- Niech $l_1 \leqslant l_2 \leqslant \ldots \leqslant l_N$.
- Definiujemy pomocnicze $w_1, w_2, \ldots, w_N$ jako
$$w_1 = 0 \qquad w_j = \sum_{i=1}^{j-1} 2^{l_j - l_i}$$
- Binarna reprezentacja $w_j$ dla $j > 1$ zajmuje $\lceil \log w_j \rceil$ bitów.
- Liczba bitów $w_j$ jest mniejsza lub równa $l_j$. Dla $w_1$ to oczywiste.
$$\begin{aligned} \log w_j &= \log \left[ \sum_{i=1}^{j-1} 2^{l_j - l_i} \right] = \log \left[ 2^{l_j} \sum_{i=1}^{j-1} 2^{-l_i} \right] = \\ &= l_j + \log \left[ \sum_{i=1}^{j-1} 2^{-l_i} \right] \leqslant l_j \end{aligned}$$



Notatki

## Konstrukcja kodu o podanych długościach

**Kodowanie wygląda następująco:**

Jeżeli $\lceil \log w_j \rceil = l_j$, to $j$-te słowo kodowe jest binarną reprezentacją $w_j$. Jeżeli jest mniejsze, to reprezentację $w_j$ uzupełniamy odpowiednią liczbą zer z lewej strony.



Notatki

## Konstrukcja kodu o podanych długościach

**Czy to jest kod prefiksowy?**

- Załóżmy, że $c_j$ jest prefiksem $c_k$ i $j < k$. Wtedy $l_j$ pierwszych bitów $c_k$ tworzy $c_j$, czyli $w_j = \left\lfloor \frac{w_k}{2^{l_k - l_j}} \right\rfloor$
- Ale $w_k = \sum_{i=1}^{k-1} 2^{l_k - l_i}$.
- Czyli

$$\begin{aligned} \frac{w_k}{2^{l_k - l_j}} &= \sum_{i=1}^{k-1} 2^{l_j - l_i} = w_j + \sum_{i=j}^{k-1} 2^{l_j - l_i} = \\ &= w_j + 2^0 + \sum_{i=j+1}^{k-1} 2^{l_j - l_i} \geqslant w_j + 1 \end{aligned}$$

- Sprzeczne z założeniem, że $c_j$ jest prefiksem $c_k$.



Notatki

## Przykład

- Weźmy $a, b, c, d$ z prawdopodobieństwami $\frac{1}{3}, \frac{1}{4}, \frac{1}{4}, \frac{1}{6}$.
- Odpowiednio długości kodów Shannon-Fano wynoszą $2, 2, 2, 3$.
- Wyliczamy $w_a = 0$, $w_b = 1$, $w_c = 2$, $w_d = 6$.
- Kody to odpowiednio
  $kod(a) = 00$, $kod(b) = 01$, $kod(c) = 10$, $kod(d) = 110$.



Notatki

Notatki

Notatki