Kompresja danych DKDA LI0

Marcin Gogolewski
marcing@wmi.amu.edu.pl

Uniwersytet im. Adama Mickiewicza w Poznaniu

Poznań, 3 października 2025

Wykład 1: Wprowadzenie do kompresji danych

Dyżury	Wtorki: 12:00-13:00
Kontakt	e-mail: marcing@wmi.amu.edu.pl, tel.: (061)8295400
Pokój	B0-15
Egzamin	egzaminy „połówkowe”
Poprawka	kolokwium z całości materiału (zadania) + egzamin
Dodatkowe punkty	za zadania „do przemyślenia” (jeżeli się pojawią), za projekty, ew. za znalezione błędy
Strona WWW	`http://marcing.faculty.wmi.amu.edu.pl/DKDA`

Literatura

- D. Karwowski, *Zrozumieć Kompresję Obrazu (ISBN 978-83-953420-0-4)*
- K. Sayood, *Kompresja danych – wprowadzenie*, READ ME 2002 (ISBN 83-7243-094-2)
- A. Drozdek *Wprowadzenie do kompresji danych*, WNT 1999 (ISBN 83-204-2303-1)
- 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)

Główne cele

- redukcja kosztów (przechowywania i przesyłu)
  np. modemy, faksy, strumienie multimedialne, ...
- maksymalne wykorzystanie dostępnych możliwości
  np. telewizja satelitarna, GSM, ...
- skrócenie czasu transmisji danych
  np. obrazów z satelity szpiegowskiego, ...

Kompresja bezstratna i stratna

Kompresja 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)

Podstawowe „prawo" kompresji *bezstratnej*

Nie istnieje algorytm, który potrafi zmniejszyć rozmiar dowolnych danych, ponieważ:

- Kompresja bezstratna jest funkcją różnowartościową.
- Różnych danych długości n jest 2^n (ciągi bitów).
- Różnych ciągów bitowych długości mniejszej niż n jest $2^n - 1$. ($\sum_{i=0}^{n-1} 2^i$ – ciąg długości zero też może być pewną informacją)
- **Wniosek 1:** istnieje ciąg, który nie może być skompresowany.
- **Wniosek 2:** ponad połowa ciągów skróci się o co najwyżej 1 bit.

Ciekawostka

Jak wykazaliśmy wcześniej

Algorytm kompresji (oraz dekompresji) bezstratnej musi być bijekcją (funkcją wzajemnie jednoznaczną). Zatem nie istnieje algorytm, który potrafi skompresować dowolny ciąg binarny!

Choć w USA został zgłoszony i zatwierdzony (!) patent na taki algorytm...

W Australii udało się nawet opatentować koło (aby wykazać, że nie można zwolnić specjalistów z urzędów patentowych).

Podstawowe metody kompresji (bezstratnej)

1. Kodowanie Huffmana (i pokrewne metody).
2. Kodowanie arytmetyczne.
3. Kodowanie słownikowe.
4. Kodowanie predykcyjne.
5. ...

A także wersje „dynamiczne” powyższych sposobów (zostaną omówione później).

Podstawowe metody kompresji (stratnej)

1. Kodowanie różnicowe (z kwantyzatorem).
2. Kodowanie transformujące (falki, FFT, DCT).
3. Kodowanie podpasmami.
4. Kodowanie fraktalne.
5. Kodowanie wideo (wiele klatek jednocześnie).
6. ...

Rozmiar „reprezentacji danych"

Liczba bitów na próbkę

Liczba bitów użyta do zakodowania pojedynczej „próbki" sygnału, np. 8 w przypadku obrazka o 256 poziomach szarości (także 8 bpp – *bit per pixel*) lub 16 w przypadku pojedynczego kanału dźwiękowego o „jakości CD".

Liczba bitów na sekundę

Miara stosowana w przypadku, gdy dane są w jakiś sposób powiązane z czasem (np. dźwięk, film). Określa liczbę bitów potrzebną do zakodowania jednej sekundy sygnału (np. $16 \times 44100 = 705600\mathrm{bps} \approx 690\mathrm{Kbps} \approx 86\mathrm{KB/s}$ w przypadku pojedynczego kanału dźwięku w „jakości CD").

Miary dla dowolnego rodzaju kompresji

Stopień kompresji

Proporcja pomiędzy danymi oryginalnymi a danymi skompresowanymi (np. 4:1, czasami także podawana jako procent: 25%).

Średnia bitowa (liczba bitów na próbkę)

Średnia (!) liczba bitów na pojedynczą próbkę (np. w przypadku obrazu „8-bitowego" o stopniu kompresji 16:3 otrzymujemy $1,5$ bita na próbkę).

Złożoność algorytmu i czas działania

Parametr ten, choć jest znaczeniowo niezależny od poprzednich, to musi być brany pod uwagę przy projektowaniu każdego systemu!

Rodzaje kodowania

Kodowanie

Przyporządkowanie elementom jakiegoś alfabetu ciągów binarnych.

Kodowanie może mieć różne cele:

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

„Ogólnie znane" sposoby kodowania

ASCII kod 7-bitowy, stała długość

UTF8 zmienna długość (1 symbol – 1-4 bajty)

Morse'a zmienna długość (często występujące litery mają krótsze kody!)

Modelowanie danych - Przykład 1

- Rozważmy 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 $e_n = x_n - \hat{x}_n$.
- Ciąg e_n przyjmuje 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.

Modelowanie danych - Przykład 2

- Rozważmy 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
- Otrzymany ciąg jest prostszy do zakodowania.

Modelowanie danych - Przykład 3

- Rozważmy tekst:
  a_barayaran_array_ran_far_faar_faaar_away.
- Mamy 8 różnych symboli, więc na każdy potrzebujemy 3 bity.
- Jeśli użyjemy kodu z tabelki, to zostaną użyte 103 bity zamiast 123 ($< 2,52$ bitu na symbol)

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

- Wykorzystujemy własności statystyczne danych.

Miary dla kompresji stratnej

„Poziom zniekształcenia danych" (*distortion*)

Różnego (!) rodzaju miary określające poziom zniekształcenia danych. Stosowane są zarówno miary „matematyczne" jak:

- średnia (bezwzględna) różnica pomiędzy próbkami:

$$\frac{1}{n}\sum_{i=1}^{n}|o_i - r_i|,$$

- różnica średniokwadratowa:

$$\frac{1}{n}\sum_{i=1}^{n}(o_i - r_i)^2,$$

jak i miary „percepcyjne" (parametry są dobierane w trakcie badań nad reprezentatywną „próbką" użytkowników – np. standard JPEG).

Średnia długość kodu

Niech $\mathcal{A}$ będzie alfabetem składającym się z N symboli $a_1, a_2, \ldots, a_N$ o prawdopodobieństwach $\Pr(a_i)$ dla $i \in \{1, 2, \ldots, N\}$ oraz długościach odpowiadających im kodów: $n(a_i)$ dla $i \in \{1, 2, \ldots, N\}$. Średnią długością kodu nazywamy takie l, że:

$$l = \sum_{i=1}^{N} \Pr(a_i) n(a_i).$$

Nierówność Krafta-McMillana

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

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

Stwierdzenie odwrotne nie jest prawdziwe!

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

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

- 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.

Nierówność Krafta-McMillana - dowód

- Sumę z poprzedniej strony 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.
- 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

$$A_k \leqslant 2^k$$

Nierówność Krafta-McMillana - dowód

- 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

$$K(C) \leqslant 1.$$

Przykład (z książki Khalida Sayooda)

Czy poniższe ciągi zawsze pozwalają na jednoznaczne odtworzenie treści:

Litera	Prawdopod.	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	0111
śr. długość:		1,125	1,25	1,75	1,875

$$\text{średnia długość} = \sum_{i=1}^{4} P(a_i)n(a_i),$$

gdzie $n(a_i)$ oznacza liczbę bitów a_i.
Wykorzystując kod 2 zakodujmy i odkodujmy ciąg $a_2a_1a_1$.
Czym różnią się kody 3 i 4?
Jaki jest algorytm sprawdzania jednoznaczności?

Kody natychmiastowe (1)

Litera	Słowo kodowe
a_1	0
a_2	01
a_3	11

Jakim symbolom odpowiada ciąg 01111111111 (10 jedynek)?

Kody natychmiastowe (2)

Kod natychmiastowy

Jest kodem pozwalającym stwierdzić w którym miejscu zakończone jest słowo kodowe w momencie odczytania ostatniej litery (litery kodu, np. 0 lub 1) tego słowa.

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*).

Kody prefiksowe

- Kod w którym żadne słowo kodowe nie jest prefiksem innego słowa kodowego.
- Łatwo zauważyć (znając algorytm sprawdzania jednoznacznej dekodowalności), że kod prefiksowy jest jednoznacznie dekodowalny.

Przykład (z książki Khalida Sayooda)

Który z poniższych kodów jest kodem prefiksowym?

Litera	Prawdopod.	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	0111
śr. długość:		1,125	1,25	1,75	1,875

Teoria informacji – *autoinformacja*

- 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
$$\begin{aligned} i(AB) &= \log \frac{1}{P(AB)} = \log \frac{1}{P(A)P(B)} = \\ &= \log \frac{1}{P(A)} + \log \frac{1}{P(B)} = i(A) + i(B) \end{aligned}$$
- Jeżeli podstawą logarytmu jest 2, to jednostką informacji jest *bit*.

Teoria informacji – entropia

- 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 tę nazywamy *entropią*.
- Kody jednoznacznie dekodowalne w modelu z niezależnymi wystąpieniami symboli muszą mieć średnią długość co najmniej równą entropii.

Przykład

- Rozważmy 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 bita na znak.
- Jeżeli jednak założymy, że elementy ciągu nie są niezależne i zastąpimy ciąg różnicami, to otrzymamy ciąg:
  1 1 1 -1 1 1 1 -1 1 1 1 1 1 -1 1 1
  (model Markova – wykorzystywany m.in. w metodach słownikowych).

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 bita na znak).

Własności optymalnych kodów prefiksowych

- Symbolom występującym częściej odpowiadają krótsze słowa kodowe.
- Dwa najrzadziej występujące symbole mają w kodzie optymalnym słowa kodowe tej samej długości.

Konstruowanie kodów Huffmana

- Kody dwóch najrzadziej występujących symboli różnią się tylko na ostatniej pozycji.
- Algorytm rekurencyjny: rozważ dwa najrzadziej występujące symbole rozróżniając je na końcu przez 0 i 1. Połącz oba w jeden symbol pomocniczy i rozważ teraz rekurencyjnie mniejszy alfabet. Powtarzaj aż zostanie tylko jeden symbol.

Przykład

- $A = \{a, b, c, d, e\}$, $P(b) = 0,4$, $P(a) = P(c) = 0,2$ i $P(d) = P(e) = 0,1$.
- Rozważamy d i e: $kod(d) = kod(x)0$, $kod(e) = kod(x)1$ i $P(x) = 0,2$.
- $A' = \{a, b, c, x\}$.
- Rozważamy c i x: $kod(c) = kod(y)0$, $kod(x) = kod(y)1$ i $P(y) = 0,4$.
- $A'' = \{a, b, y\}$.
- Rozważamy a i b: $kod(a) = kod(z)0$, $kod(b) = kod(z)1$ i $P(z) = 0,6$.
- $A''' = \{y, z\}$.
- Rozważamy y i z: $kod(y) = 0$ i $kod(z) = 1$.
- Rekonstruujemy kody: $kod(a) = 10$, $kod(b) = 11$, $kod(c) = 00$, $kod(d) = 010$, $kod(e) = 011$.

Własności kodów Huffmana

- Kod Huffmana jako drzewo – tworzenie takiego drzewa jako algorytm tworzenia kodów Huffmana.
- Niejednoznaczność tworzenia kodów – istnieje często wiele kodów dla jednego źródła danych (ale wszystkie mają tą samą średnią długość kodu!).

Optymalność kodów Huffmana

Optymalny kod powinien spełniać następujące warunki:

- Dla każdych dwóch liter a i b takich, że $P(a) \geqslant P(b)$ zachodzi $l_b \geqslant l_a$.
- Dwie litery o najmniejszych prawdopodobieństwach mają słowa kodowe o tej samej, maksymalnej długości.
- Z każdego wierzchołka wewnętrznego drzewa odpowiadającego kodowi optymalnemu powinny wychodzić oba poddrzewa.
- Jeśli połączymy dwa liście mające wspólnego ojca i ten wierzchołek wewnętrzny potraktujemy jako liść to uzyskane drzewo jest kodem optymalnym dla nowego alfabetu, jeśli pierwotne drzewo było optymalne.

Długość kodów Huffmana

Dla źródła S spełniona jest nierówność

$$H(S) \leqslant l \leqslant H(S) + 1$$

gdzie l - średnia długość kodu Huffmana dla źródła S.

Uwaga: entropia jest dobrym oszacowaniem na średnią długość kodu w przypadku modelu z niezależnością zdarzeń.

Rozszerzone i niebinarne kody Huffmana

- Kody Huffmana możemy zastosować także do bloków symboli.
- Kody Huffmana można tworzyć także dla innego niż binarny alfabetu wyjściowego – prosta modyfikacja algorytmu: zamiast dwóch symboli łączymy m symboli – niestety alfabet rośnie wykładniczo ze względu na m...

Podsumowanie – własności kodów Huffmana

- Optymalność wśród kodów prefiksowych.
- Kodowanie i dekodowanie w czasie liniowym (szybkie).
- Kody rozszerzone - kompromis między optymalnością a wielkością systemu.
- Możliwość implementacji dynamicznej.

Zastosowanie: pkZIP, lha, gzip, zoo, arj, fragmenty formatów JPEG i MPEG.