Kody Tunstalla. Kodowanie arytmetyczne

Kodowanie i kompresja danych - Wykład 3

Maciek Gębala

11 marca 2020

Notatki

Kody Tunstalla

- Wszystkie słowa kodowe mają tą samą długość, ale jeden kod może kodować różną liczbę liter alfabetu wejściowego.
- Chcemy zmaksymalizować średnią liczbę symboli z pierwotnego alfabetu reprezentowanych przez słowa kodowe.
- Alfabet wejściowy: litery $a_1, a_2, \ldots, a_N$ z prawdopodobieństwami $p_1, p_2, \ldots, p_N$ (N symboli).
- Kody są długości n bitów.

Notatki

Kody Tunstalla

Algorytm tworzenia kodów Tunstalla

- Przyporządkowujemy symbolom alfabetu N różnych słów kodowych (o długości n).
- Dopóki liczba niewykorzystanych słów kodowych jest większa niż $N - 1$:
  - wybierz słowo kodowe e odpowiadające ciągowi o największym prawdopodobieństwie;
  - usuń e z kodu;
  - dodaj do kodu ciągi powstałe z dodania $a_1, \ldots, a_N$ jako sufiksów ciągu odpowiadającego kodowi e (przypisz im odpowiednie prawdopodobieństwa).

Co najmniej jedno słowo kodowe zostanie niewykorzystane.

Notatki

Przykład

- Weźmy alfabet a, b, c ($N = 3$) z prawdopodobieństwami $P(a) = 0.6$, $P(b) = 0.3$ i $P(c) = 0.1$.
- Ustalmy $n = 3$ (8 słów kodowych).
- Mamy 3 słowa kodowe odpowiadające $a \to 0.6$, $b \to 0.3$ i $c \to 0.1$.
- Zastępujemy a i otrzymujemy 5 kodów: $aa \to 0.36$, $ab \to 0.18$, $ac \to 0.06$, $b \to 0.3$ $c \to 0.1$.
- Zastępujemy aa i otrzymujemy 7 kodów: $aaa \to 0.216$, $aab \to 0.108$, $aac \to 0.036$, $ab \to 0.18$, $ac \to 0.06$, $b \to 0.3$ i $c \to 0.1$.

aaa	aab	aac	ab	ac	b	c	???
000	001	010	011	100	101	110	111

Notatki

Przykład

aaa	aab	aac	ab	ac	b	c	???
000	001	010	011	100	101	110	111

- Zakodujmy tekst *abcaabbaa*.
- Otrzymujemy: 001110001101??.
- Na końcu tekstu może pojawić się blok dla którego nie ma słowa kodowego, wtedy wysyłamy specjalny kod i normalne kody liter.
- 001110001101111*kod*(*a*)*kod*(*a*)

Notatki

Średnia długość

- Średnia liczba bitów na jeden symbol wejściowy.

$$\sum_{i=1}^{2^n-1} P(e_i)\frac{n}{|e_i|},$$

gdzie e_i - słowo odpowiadające i-temu kodowi.
- Dla przykładu z poprzedniego slajdu: $(0.216+0.108+0.036)\frac{3}{3}+$ $(0.18+0.06)\frac{3}{2}+$ $(0.3+0.1)\frac{3}{1}=1.92$
- Średnia długość kodu Huffmana dla tego przypadku to 1.4.

Notatki

Kody Tunstalla - podsumowanie

- Zmienna długość bloków wejściowych, stała długość wyjściowych.
- Kompresja i odporność na przekłamania.
- Jednoznaczność kodowania i dekodowania.

Notatki

Kodowanie arytmetyczne

- Tekst wejściowy zostaje odwzorowany na liczbę z przedziału $[0,1)$.
- Kod tekstu to liczba n - długość tekstu i liczba z (znacznik) reprezentowany z odpowiednio dobraną dokładnością.
- Elementom alfabetu $a_1,\ldots,a_N$ z prawdopodobieństwami $p_1,\ldots,p_N$ przyporządkowujemy przedziały $[F(i),F(i+1))$, gdzie $F(i)=\sum_{j=1}^{i-1} p_i$.

Kodowanie

- Mamy ciąg liter $x_1x_2\ldots x_m$ z alfabetu $a_1,\ldots,a_N$.
- Na początku przedział $[l,p)=[0,1)$.
- Dla $i=1,2,\ldots,m$:
  - Niech $x_i=a_j$.
  - Wtedy $d\leftarrow p-l$, $p\leftarrow l+F(j+1)d$ i $l\leftarrow l+F(j)d$.
- Znacznik to dowolna liczba z przedziału $[l,p)$, np. $z=(l+p)/2$.

Notatki

Przykład

- Weźmy alfabet a, b, c z prawdopodobieństwami $0.7, 0.1, 0.2$.
- Zakodujmy tekst *abc*.
- Na początku mamy przedział: $[0, 1)$.
- $F(1) = 0$, $F(2) = 0.7$, $F(3) = 0.8$, $F(4) = 1$.
- Kodujemy a i otrzymujemy przedział $[0, 0.7)$.
- Kodujemy b i otrzymujemy przedział $[0.49, 0.56)$.
- Kodujemy c i otrzymujemy przedział $[0.546, 0.56)$.
- Za znacznik możemy przyjąć 0.553.

Notatki

Kodowanie arytmetyczne

Dla ustalonej długości tekstu n, każdy ciąg jest odwzorowany na przedział rozłączny z przedziałami odpowiadającymi innym ciągom. Gwarantuje to jednoznaczność kodowania.

Wygenerowanie znacznika dla konkretnego ciągu nie wymaga wyznaczania bądź pamiętania znaczników innych ciągów.

Dekodowanie

- Dostajemy n - długość tekstu i z - znacznik tekstu.
- $l \leftarrow 0$ i $p \leftarrow 1$.
- Dla $i = 1, 2, \ldots, n$:
  - Wybieramy j takie, że $l + F(j)(p - l) \leqslant z < l + F(j + 1)(p - l)$;
  - Przyjmujemy, że $x_i = a_j$;
  - $d \leftarrow p - l$, $p \leftarrow l + F(j + 1)d$ i $l \leftarrow l + F(j)d$.
- Ciąg oryginalny to $x_1, \ldots, x_n$.

Notatki

Przykład

- $P(a) = 0.7$, $P(b) = 0.1$, $P(c) = 0.2$, $z = 0.55$ i $n = 3$.
- $F(1) = 0$, $F(2) = 0.7$, $F(3) = 0.8$ i $F(4) = 1$.
- $l = 0$ i $p = 1$.
- Dla a mamy $0 \leqslant 0.55 < 0.7$, stąd $x_1 = a$, $l = 0$ i $p = 0.7$.
- Dla b mamy $0.49 \leqslant 0.55 < 0.56$, stąd $x_2 = b$, $l = 0.49$ i $p = 0.56$.
- Dla c mamy $0.546 \leqslant 0.55 < 0.56$, stąd $x_3 = c$, $l = 0.546$ i $p = 0.56$.
- Odkodowany ciąg to *abc*

Notatki

Własności

Jak reprezentować znacznik (liczba rzeczywista) aby był jak najkrótszy

- Niech $x = x_1, \ldots, x_n$ będzie ciągiem danych o prawdopodobieństwie wystąpienia $P(x) = \Pi_{i=1}^{n} P(x_i)$. Zaokrąglenie z' znacznika z do

$$m(x) = \left\lceil \log \frac{1}{P(X)} \right\rceil + 1$$

bitów gwarantuje jednoznaczność kodowania.
- Kod arytmetyczny dla ustalonej długości tekstu jest kodem prefiksowym.

Notatki

Przykład

- $P(a) = 0.7$, $P(b) = 0.1$, $P(c) = 0.2$.
- Kod dla tekstu *abc* to $0.553 = (0.100011011)_2$.
- $P(abc) = 0.014$.
- $\lceil \log \frac{1}{0.014} \rceil + 1 = 8$
- Czyli do zakodowania tekstu wystarczy wysłać 10001101.

Notatki

Usprawnienia

- Dla długich ciągów potrzebujemy długich liczb i przetwarzanie wymaga przeczytania całego ciągu.
- Można zmodyfikować algorytm do pracy przyrostowej - znacznik powstaje etapami i można wysyłać go fragmentami.

Notatki

Kodowanie ze skalowaniem

Po zakodowaniu kolejnej litery:

- Jeśli $[l, p) \subseteq [0, 0.5)$:
  - zamień $[l, p)$ na $[2l, 2p)$;
  - dołącz do kodu słowo $01^{licznik}$;
  - $licznik \leftarrow 0$.
- Jeśli $[l, p) \subseteq [0.5, 1)$:
  - zamień $[l, p)$ na $[2l - 1, 2p - 1)$;
  - dołącz do kodu słowo $10^{licznik}$;
  - $licznik \leftarrow 0$.
- Jeśli $l < 0.5 < p$ i $[l, p) \subseteq [0.25, 0.75)$:
  - zamień $[l, p)$ na $[2l - 0.5, 2p - 0.5)$;
  - $licznik \leftarrow licznik + 1$.

Analogicznie można zmodyfikować procedurę dekodowania aby dekodowanie odbywało się na podstawie otrzymywanych fragmentów.

Notatki

Dalsze możliwe usprawnienia

- Przejście z arytmetyki zmiennoprzecinkowej na arytmetykę całkowitoliczbową: zastąpienie przedziału $[0, 1)$ przez przedział liczb całkowitych $[0, 2^m - 1]$.
- Problem: Jak dobrać m aby uniknąć błędów zaokrągleń.

Notatki

Kodowanie arytmetyczne – algorytm całkowitoliczbowy

- Odwzorowujemy istotne wartości z przedziału $[0, 1)$ na zbiór 2^m wartości binarnych.
  - odwzorowujemy 0 na $\overbrace{000\ldots0}^{m \text{ razy}}$
  - odwzorowujemy 1 na $\overbrace{111\ldots1}^{m \text{ razy}}$
  - odwzorowujemy $0,5$ na $1\overbrace{00\ldots0}^{m-1 \text{ razy}}$
- Zamieniamy wyrażenia aktualizujące tak, aby uwzględniały zaokrąglenia.
- Skalowanie działa tak, jak w przypadku oryginalnego algorytmu.

Notatki

Porównanie kodowania arytmetycznego i Huffmana

Niech $H(S)$ oznacza entropię źródła S, l_A średnią długość kodu arytmetycznego, l_H średnią długość kodu Huffmana a m długość bloku:

$$H(S) \leqslant l_A \leqslant H(S) + \frac{2}{m}$$

oraz

$$H(S) \leqslant l_H \leqslant H(S) + \frac{1}{m}$$

W przypadku kodowania Huffmana teoretyczna wartość jest mała, jednak dla dużych wartości m kody Huffmana są niepraktyczne.

Notatki

Notatki

Notatki