Spis treści

1 Wstęp

Wyróżniamy dwa rodzaje kompresji. W kompresji stratnej dopuszczalny jest pewien stopień straty informacji wejściowej. W kompresji bezstratnej nie jest to dopuszczalne.

1.1 "Prawo" Kompresji bezstratnej

Nie istnieje algorytm, który potrafi zmniejszyć rozmiar dowolnych danych

- Kompresja bezstratna musi być bijekcją
- Dowolne dane przyjmują postać ciągu bitów długości n. Jest 2^n takich ciągów.
- Danych krótszych niż n, np.: o jeden jest 2^{n-1}
- Nie da się stworzyć bijekcji z zbioru o mocy 2^n do zbioru o mocy 2^{n-1}

Wniosek jest taki, że koniecznym jest konstruowanie kompresji bezstratnej na podzbiorach danych, takich jak np.: obrazów, dźwięków, tekstów.

2 Kodowania

Kodowanie to przyporządkowanie elementom jakiegoś alfabetu ciągu binarnych. Przykładami kodowania są: ASCII, UTF-8 oraz inne. Typowym jest konstruowanie kodowania pod konkretny zestaw danych, optymalizując je pod kątem częstości występowania poszczególnych elementów.

2.1 Modelowanie danych

Rozważmy ciąg: $a_n = 9, 11, 11, 11, 14, 13, 15, 17, 16, 17, 20, 21$. $\max(a_n) = 21$ stąd koniecznym jest 5 bitów na element. Ale jeśli wykorzystamy wzór $e_n = a_n - n + 8$ do stworzenia nowego ciągu, to ten ciąg przyjmuje postać: $0, 1, 0, -1, 1, -1, 0, 1, -1, -1, 1, 1$. Teraz wystarczą tylko 2 bity na zakodowanie elementu.

2.2 Średnia długość kodu

$$I = \sum_{i=1}^{n} p_i \cdot l_i$$

gdzie p_i to prawdopodobieństwo wystąpienia elementu i, a l_i to długość kodu dla elementu i.

2.3 Jednoznaczna dekodowalność

Jeśli dla dowolnego ciągu znaków istnieje tylko jedno jego rozkodowanie to kod jest jednoznacznie dekodowalny. Aby sprawdzić czy kod jest jednoznacznie dekodowalny, należy zastosować następujący algorytm.

1. Stwórz pustą listę
2. Dla każdej pary słów kodowych sprawdź czy jedno jest prefiksem drugiego. Jeśli tak, dodaj sufiks drugiego słowa do listy, jeśli już go tam nie ma.
3. Jeśli na liście jest słowo kodowe, to kod nie jest jednoznacznie dekodowalny.

2.3.1 Nierówność Krafta

Jeżeli $\mathcal{C}$ jest kodem jednoznacznie dekodowalnym z n słowami to:

$$K(\mathcal{C}) = \sum_{i=1}^{n} 2^{-l_i} \leq 1$$

Jest to warunek konieczny bycia kodem jednoznacznie dekodowalnym.

2.3.2 Kod prefiksowy

Kod w którym żadne słowo kodowe nie jest prefiksem innego słowa kodowego. Wszystkie kody prefiksowe są jednoznacznie dekodowalne.

2.4 Kod natychmiastowy

Jest kodem pozwalającym stwierdzić w którym miejscu zakończone jest słowo kodowe w momencie odczytania ostatniej litery.

2.5 Statyczny Kod Huffmana

Kod Huffmana to kod prefiksowy o minimalnej średniej długości kodu. Są one optymalne wśród kodów prefiksowych.

Dla alfabetu $\mathcal{A}$ o długości n i prawdopodobieństwach wystąpienia $p_1, \ldots, p_n$ algorytm tworzenia kodu Huffmana wygląda następująco: Znajdź dwa najrzadziej występujące elementy i połącz je w jeden element o prawdopodobieństwie $p_1 + p_2$. Rozróżnij je 0 lub 1. Powtórz ten krok na liście $n - 1$ długiej aż zostanie jeden element.

2.6 Kodowanie Shannon-Fano

Dla symboli $a_1, \ldots, a_n$ o prawdopodobieństwach $p_1, \ldots, p_n$, ustalmy kody długości $l_n = \lceil -\log p_i \rceil$. Następnie zdefiniujmy zmienne pomocnicze $w_1, \ldots w_n$ jako:

$$w_1 = 0, w_j = \sum_{i=1}^{j-1} 2^{l_j - l_i}$$

Jeżeli $\lceil \log w_j \rceil = l_j$ to j-te słowo kodowe jest binarną reprezentacją w_j. Jeżeli $\lceil \log w_j \rceil < l_j$ to reprezentację uzupełniamy zerami z lewej strony.

Dla $P(a) = \frac{1}{3}, P(b) = \frac{1}{4}, P(c) = \frac{1}{4}, P(d) = \frac{1}{6}$ mamy:

$$l_a = 2, l_b = 2, l_c = 2, l_d = 3$$

$$w_1 = 0, w_2 = 2, w_3 = 2, w_4 = 6$$

$$kod(a) = 00, kod(b) = 01, kod(c) = 10, kod(d) = 110$$

2.7 Kodowanie Tunstalla

Chcemy stworzyć kod na n bitach dla $a_1, \ldots, a_m$ symboli o prawdopodobieństwach $p_1, \ldots, p_m$. Tworzenie kodu Tunstalla polega na iteracyjnym wyborze ze zbioru symbolu o największym prawdopodobieństwie S i łączenie go z wszystkimi innymi symbolami tworząc symbole Sa_m, nadając im prawdopodobieństwa $P \cdot p_m$. Proces ten powtarzamy aż do uzyskania kodu o długości n.

2.8 Kodowanie Golomba

Kody Golomba są parametryzowane liczbą $m > 0$. Każda liczba n jest zapisywana za pomocą $q = \lfloor \frac{n}{m} \rfloor$ oraz $r = n - q \cdot m$ w postaci

$$(q)_1(r)_2$$

2.9 Dynamiczne kodowanie Huffmana

3 Teoria informacji

Teoria informacji to dziedzina zajmująca się przetwarzaniem informacji.

3.1 Miara informacji

Miarą informacji, którą niesie ze sobą zdarzenie A jest:

$$I(A) = -\log_x P(A)$$

gdzie x to baza systemu liczbowego. Jeśli miarą informacji jest bit to $x = 2$. Jeśli zdarzenia A i B są niezależne to:

$$I(AB) = I(A) + I(B)$$

3.2 Entropia

Entropia to miara średniej informacji przekazywanej przez źródło. Kody jednoznacznie dekodowalne w modelu z niezależnymi wystąpieniami symboli muszą mieć średnią długość co najmniej równą entropii.

3.2.1 Entropia źródła

Dla źródła danych S generującego ciąg X nad alfabetem $\mathcal{A} = \{1, 2, \ldots m\}$

$$H(S) = \lim_{n \to \infty} \frac{G_n}{n}$$

$$G_n = -\sum_i \cdots \sum_j P(X_1 = i, \ldots, X_n = j) \log P(X_1 = i, \ldots, X_n = j)$$

3.2.2 Entropia Pierwszego Rzędu

Dla źródła informacji X, z zbiorem wiadomości (zdarzeń) $A_1, \ldots, A_n$, gdzie $P(A_i)$ to prawdopodobieństwo wystąpienia zdarzenia A_i i zdarzenia są niezależne to entropia źródła to:

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

4 Kodowanie uniwersalne

Szukamy sposobu na kodowanie dowolnej liczby $n \in \mathbb{N}$. Problem polega na skonstruowaniu kodu, który będzie jednoznacznie dekodowalny i uniwersalny. To oznacza, że ma się skalować w nieskończoność.

4.1 Kodowanie Eliasa

Kodowanie Eliasa to kodowanie uniwersalne, które wykorzystuje kodowanie unarne do zapisania długości kodu binarnego liczby n.

$$n = \lfloor \log_2(x) \rfloor + 1$$

4.1.1 γ

Jest to najprostsze z kodowań Eliasa. Polega na zakodowaniu liczby x w postaci binarnej, a następnie dodaniu przed nią liczby $n-1$ zer.

$$\gamma(x) = 0^{n-1}(x)_2$$

$$(13)_{10} = 1101_2 \Rightarrow \gamma(13) = 0001101$$

4.1.2 δ

Cały trik kodu δ polega na zakodowaniu długości kodu binarnego liczby x przy pomocy kodu γ. Istotnym trikiem jest usunięcie najstarszego bitu z zakodowanej liczby x.

$$\delta(x) = \gamma(n) + (x)_2$$

$$(13)_{10} = 1101_2 \Rightarrow \delta(13) = 00100101$$

Jak widać, jest on bardziej efektywny dla większych liczb. Długość kodu δ to $2 \cdot \lceil \log_2(\lceil \log_2 x \rceil) \rceil - 1 + \lceil \log_2 x \rceil - 1$.

4.1.3 ω

Jest to kodowanie rekurencyjne, które działa jak kodowanie δ, ale w nieskończoność. Na koniec umieszczane jest 0, potem kodowana jest liczba $k = x$. Potem ten krok jest powtarzany dla $k = n - 1$ gdzie n to liczba bitów z poprzedniego kroku.

$$(13)_{10} = 1101_2 \Rightarrow \omega(13) = 1111010$$

4.2 Kodowanie Fibonacciego

Liczba Fibonacciego ma postać:

$$f_0 = f_1 = 1$$

$$f_n = f_{n-1} + f_{n-2} : n \geq 2$$

Kodowanie fibonacciego polega na reprezentacji liczby x jako sumę liczb fibonacciego.

$$x = \sum_{i=0} a_i \cdot f_i, a_i \in \{0, 1\}$$

$$(13)_{10} = f_7 = 1101_2 \Rightarrow Fib(13) = 0000011$$