Kodowanie Arytmetyczne

Tomasz Danel

Teoria Informacji, 2016

Plan

Kompresja Danych

- Kompresja danych jest możliwa, gdy symbole tekstu występują w nim z różnymi prawdopodobieństwami.
- Shannon pokazał, że dla kodowania optymalnego symbol, którego prawdopodobieństwo wystąpienia wynosi p, jest kodowany przez $-\log_2 p$ bitów.
- Kodowanie arytmetyczne zbliża się do tej liczby.

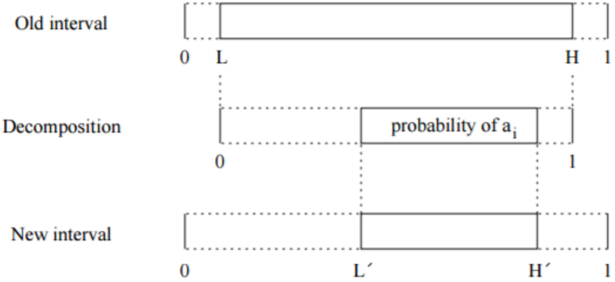
Kodowanie Arytmetyczne

Zalety i Wady

- Elastyczność,
- Optymalność,
- Powolność,
- Nieprefiksowość kodu,
- Podatność na błędy.

Plan

Algorytm Podstawowy



Rysunek: Podział bieżącego przedziału bazujący na prawdopodobieństwie symbolu a_i, który wystąpi w tekście jako następny.

Algorytm Podstawowy

Kroki

1. Zaczynamy z przedziałem $[L, H) = [0, 1)$.
2. Dla każdego symbolu w pliku wykonujemy następujące kroki:
   2.1 Dzielimy przedział na podprzedziały, po jednym dla każdego symbolu alfabetu. Długość przedziału jest wyestymowanym prawdopodobieństwem, że dany symbol wystąpi następny w tekście.
   2.2 Wybieramy podprzedział odpowiadający symbolowi, który jest następnym symbolem w pliku, i zmieniamy bieżący przedział.
3. Wypisujemy wystarczającą liczbę bitów, które jednoznacznie odróżniają przedział końcowy od pozostałych możliwych przedziałów.

Algorytm Podstawowy

Kroki

1. Zaczynamy z przedziałem $[L, H) = [0, 1)$.
2. Dla każdego symbolu w pliku wykonujemy następujące kroki:
   2.1 Dzielimy przedział na podprzedziały, po jednym dla każdego symbolu alfabetu. Długość przedziału jest wyestymowanym prawdopodobieństwem, że dany symbol wystąpi następny w tekście.
   2.2 Wybieramy podprzedział odpowiadający symbolowi, który jest następnym symbolem w pliku, i zmieniamy bieżący przedział.
3. Wypisujemy wystarczającą liczbę bitów, które jednoznacznie odróżniają przedział końcowy od pozostałych możliwych przedziałów.

Algorytm Podstawowy

Kroki

1. Zaczynamy z przedziałem $[L, H) = [0, 1)$.
2. Dla każdego symbolu w pliku wykonujemy następujące kroki:
   2.1 Dzielimy przedział na podprzedziały, po jednym dla każdego symbolu alfabetu. Długość przedziału jest wyestymowanym prawdopodobieństwem, że dany symbol wystąpi następny w tekście.
   2.2 Wybieramy podprzedział odpowiadający symbolowi, który jest następnym symbolem w pliku, i zmieniamy bieżący przedział.
3. Wypisujemy wystarczającą liczbę bitów, które jednoznacznie odróżniają przedział końcowy od pozostałych możliwych przedziałów.

Algorytm Podstawowy

Kroki

1. Zaczynamy z przedziałem $[L, H) = [0, 1)$.
2. Dla każdego symbolu w pliku wykonujemy następujące kroki:
   - 2.1 Dzielimy przedział na podprzedziały, po jednym dla każdego symbolu alfabetu. Długość przedziału jest wyestymowanym prawdopodobieństwem, że dany symbol wystąpi następny w tekście.
   - 2.2 Wybieramy podprzedział odpowiadający symbolowi, który jest następnym symbolem w pliku, i zmieniamy bieżący przedział.
3. Wypisujemy wystarczającą liczbę bitów, które jednoznacznie odróżniają przedział końcowy od pozostałych możliwych przedziałów.

Algorytm Podstawowy

Kroki

1. Zaczynamy z przedziałem $[L, H) = [0, 1)$.
2. Dla każdego symbolu w pliku wykonujemy następujące kroki:
   2.1 Dzielimy przedział na podprzedziały, po jednym dla każdego symbolu alfabetu. Długość przedziału jest wyestymowanym prawdopodobieństwem, że dany symbol wystąpi następny w tekście.
   2.2 Wybieramy podprzedział odpowiadający symbolowi, który jest następnym symbolem w pliku, i zmieniamy bieżący przedział.
3. Wypisujemy wystarczającą liczbę bitów, które jednoznacznie odróżniają przedział końcowy od pozostałych możliwych przedziałów.

Algorytm Podstawowy

Komentarz

- Długość ostatniego przedziału jest równy iloczynowi prawdopodobieństw poszczególnych symboli.
- Ostatni krok używa prawie dokładnie $-\log_2 p$ symboli, by rozróżnić plik od innych plików.
- Potrzebny jest mechanizm do sygnalizowania końca pliku - symbol końca pliku lub długość pliku.
- W drugim kroku liczymy tylko podprzedział odpowiadający kolejnemu symbolowi wejścia a_i przy pomocy prawdopodobieństwa kumulatywnego $P_C = \sum_{k=1}^{i-1} p_k$, $P_N = \sum_{k=1}^{i} p_k$. Nowy przedział to $[L + P_C(H - L), L + P_N(H - L))$.

Algorytm Podstawowy

Przykład 1

Zakodujmy tekst **bbb** pochodzący z pliku, z estymowanymi prawdopodobieństwami znaków $p_a = 0.4$, $p_b = 0.5$ oraz $p_{EOF} = 0.1$.

Kodowanie przebiega następująco:

Obecny Przedział	Akcja	a	b	EOF	Wejście
$[0.000, 1.000)$	Podziel	$[0.000, 0.400)$	$[0.400, 0.900)$	$[0.900, 1.000)$	b
$[0.400, 0.900)$	Podziel	$[0.400, 0.600)$	$[0.600, 0.850)$	$[0.850, 0.900)$	b
$[0.600, 0.850)$	Podziel	$[0.600, 0.700)$	$[0.700, 0.825)$	$[0.825, 0.850)$	b
$[0.700, 0.825)$	Podziel	$[0.700, 0.750)$	$[0.750, 0.812)$	$[0.812, 0.825)$	EOF
$[0.812, 0.825)$					

Algorytm Podstawowy

Przykład 1

- Końcowy przedział to $[0.8125, 0.825)$, co binarnie jest równe około $[0.1101000000, 0.1101001100)$.
- Prawdopodobieństwo tego tekstu wynosi $(0.5)^3(0.1) = 0.0125$, więc długość kodu powinna wynosić $-\log_2 p = 6.322$.
- Wypisujemy na wyjście **1101000**.

Algorytm Szczegółowy

Podstawowa implementacja ma następujące wady:

- Zmniejszanie przedziału wymaga arytmetyki wysokiej precyzji.
- Nie jest wyświetlane żadne wyjście, dopóki algorytm się nie zakończy.

Rozwiązanie

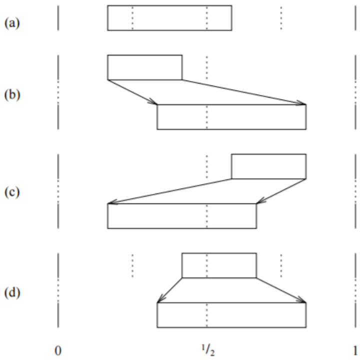
Wyświetl wiodący bit, jak tylko jest znany, a następnie podwój długość przedziału, który opisuje nieznaną część ciągu.

Kolejny Problem

Co jeśli końce przedziału ciągle znajdują się po różnych stronach 1/2, lecz dość blisko?

Algorytm Szczegółowy

Schemat



Rysunek: Podwajanie przedziału. (a) Bez podwajania. (b) Przedział w [0,1/2). (c) Przedział w [1/2, 1). (d) Przedział w [1/4,3/4).

Algorytm Szczegółowy

Trochę Kodu

```
while True:
    if interval[1] < Fraction(1, 2):
        code += '0' + follow * '1'
        interval = (interval[0] * 2, interval[1] * 2)
        follow = 0
    elif interval[0] >= Fraction(1, 2):
        code += '1' + follow * '0'
        interval = (interval[0] * 2 - 1, interval[1] * 2 - 1)
        follow = 0
    elif interval[0] >= Fraction(1, 4) and interval[1] < Fraction(3, 4):
        follow += 1
        interval = (interval[0] * 2 - Fraction(1, 2),
                    interval[1] * 2 - Fraction(1, 2))
    else:
        break
```

Algorytm Szczegółowy

Przykład 2

Zakodujemy tekst z przykładu 1 algorytmem szczegółowym.

Przedział	Akcja	a	b	EOF	Wejście
$[0.00, 1.00)$	Podziel	$[0.00, 0.40)$	$[0.40, 0.90)$	$[0.90, 1.00)$	b
$[0.40, 0.90)$	Podziel	$[0.40, 0.60)$	$[0.60, 0.85)$	$[0.85, 0.90)$	b
$[0.60, 0.85)$	Wypisz 1				
	Podwój $[\frac{1}{2}, 1)$				
$[0.20, 0.70)$	Podziel	$[0.20, 0.40)$	$[0.40, 0.65)$	$[0.65, 0.70)$	b
$[0.40, 0.65)$	follow				
	Podwój $[\frac{1}{4}, \frac{3}{4})$				
$[0.30, 0.80)$	Podziel	$[0.30, 0.50)$	$[0.50, 0.75)$	$[0.75, 0.80)$	EOF

Algorytm Szczegółowy

Przykład 2

Przedział	Akcja
$[0.75, 0.80)$	Wypisz 10
	Podwój $[\frac{1}{2}, 1)$
$[0.50, 0.60)$	Wypisz 1
	Podwój $[\frac{1}{2}, 1)$
$[0.00, 0.20)$	Wypisz 0
	Podwój $[0, \frac{1}{2})$
$[0.00, 0.40)$	Wypisz 0
	Podwój $[0, \frac{1}{2})$
$[0.00, 0.80)$	Wypisz 0

- Ponownie otrzymujemy ciąg bitów **1101000**.
- Stan końcowy kodera to $[0, 0.8)$, co zawiera $-\log_2 0.8 \approx 0.322$ bitów informacji.

Arytmetyka Liczb Całkowitych

- W praktyce przedziały możemy reprezentować jako odpowiednio duże liczby naturalne.
- Będziemy używać liczności symboli do estymowania prawdopodobieństw. $C = \sum_{k=1}^{i-1} c_k$, $N = \sum_{k=1}^{i} c_k$, $T = \sum_{k=1}^{n} c_k$ - suma wszystkich liczności.
- Nowy przedział ma postać: $\left[L + \left\lfloor \frac{C(H-L)}{T} \right\rfloor, L + \left\lfloor \frac{N(H-L)}{T} \right\rfloor\right)$.

Arytmetyka Liczb Całkowitych

Teoria

Twierdzenie 1

Jeśli używamy liczb naturalnych z zakresu $[0, N)$ *i algorytmu wysokiej precyzji do skalowania podzakresów, to można udowodnić, że długość kodu jest ograniczona przez* $4/(N \ln 2)$ *bity na symbol wejścia więcej niż idealna długość kodu dla pliku.*

Twierdzenie 2

Użycie specjalnego znaku końca pliku przy kodowaniu tekstu o długości t *przy pomocy liczb naturalnych z zakresu* $[0, N)$ *skutkuje w nadmiarze długości kodu mniejszym niż* $8t/(N \ln 2) + \log N + 7$ *bitów.*

Arytmetyka Liczb Całkowitych

Więcej Teorii

Możemy policzyć nadmiarową długość kodu, gdy założymy model z prawdopodobieństwami symboli q_i, podczas gdy prawdziwe prawdopodobieństwa to p_i. Średnia długość takiego kodu, to

$$L = -\sum_{i=1}^{n} p_i \log q_i .$$

Dla porównania optymalna długość kodu to entropia

$$H = -\sum_{i=1}^{n} p_i \log p_i ,$$

a nadmiarowość to $E = L - H$.

Arytmetyka Liczb Całkowitych

Więcej Teorii

Możemy teraz oznaczyć $d_i = q_i - p_i$ i rozszerzyć E asymptotycznie względem d

$$E = \sum_{i=1}^{n} \left(\frac{1}{2 \ln 2} \frac{d_i^2}{p_i} + O\left(\frac{d_i^3}{p_i^2} \right) \right).$$

Wniosek

Zniknięcie składników liniowych oznacza, że małe błędy w estymacji prawdopodobieństw wnoszą bardzo małą ilość nadmiarowego kodu.

Plan

Szczególny Przypadek Kodera Arytmetycznego

- Algorytm dla alfabetu wielosymbolowego można również zastosować do przypadku binarnego.
- Warto wyróżnić kodery binarne, bo są prostsze i używają prostszego dostępu do modelu.
- Popularnym problemem jest kodowanie dwupoziomowych obrazów, które generują prawdopodobieństwa bliskie 1. Popularny jest Q-Coder.

Uproszczenie Kodera Wielosymbolowego

- Możemy zbudować drzewo binarne.
- Nie trzeba tworzyć prawdopodobieństw kumulatywnych.
- W każdym węźle zmieniamy jeden koniec przedziału.
- Kodujemy wiele zdarzeń.
- Problem z przechowywaniem drzew.

Plan

Dwa Podejścia do Kodowania Tekstu

- Model 2-przejściowy:
  - Podczas pierwszego przejścia wybieramy model.
  - Podczas drugiego przejścia kodujemy tekst.
- Model 1-przejściowy (adaptacyjny):
  - Od razu kodujemy tekst i poprawiamy model w miarę wczytywania znaków.
  - Możemy założyć początkowo, że wszystkie symbole występują jeden raz.

Skalowanie

Liczności symboli w modelu adaptacyjnym mogą urosnąć do ogromnych rozmiarów, dlatego można co jakiś czas odejmować ustaloną liczbę od wszystkich liczności. Wprowadza to lokalność. (restartowanie modelu, przesuwane okno, starzenie)

Bardziej Wyszukane Modele

Fakt

Jedyną metodą osiągnięcia znacznej poprawy kompresji jest użycie bardziej wyszukanego modelu.

Spostrzeżenie

To "wyszukanie" może zostać osiągnięte przez predykcję symboli na podstawie ich kontekstu.

Prediction by Partial Matching

- Przechowujemy modele wysokich rzędów.
- Zawsze używamy modelu najwyższego dostępnego rzędu.
- Wprowadzamy symbol wyjścia.

Plan

Bardziej Wyszukane Modele

- Poprawa algorytmu Ziv-Lempel.
- Bezstratna kompresja obrazów.
- Generowanie rozkładów.

Plan

Idea

		0 input		**1** input	
State	Prob{**0**}	Output	Next state	Output	Next state
$[0,4)$	$0 < p < 1-\alpha$	**00**	$[0,4)$	-	$[1,4)$
	$1-\alpha \le p \le \alpha$	**0**	$[0,4)$	**1**	$[0,4)$
	$\alpha < p < 1$	-	$[0,3)$	**11**	$[0,4)$
$[0,3)$	$0 < p < 1/2$	**00**	$[0,4)$	*follow*	$[0,4)$
	$1/2 \le p < 1$	**0**	$[0,4)$	**10**	$[0,4)$
$[1,4)$	$0 < p < 1/2$	**01**	$[0,4)$	**1**	$[0,4)$
	$1/2 \le p < 1$	*follow*	$[0,4)$	**11**	$[0,4)$

Idea

Prościej

		0 input		1 input	
State	Prob{0}	Output	Next state	Output	Next state
$[0,4)$	$0 < p < 1-\alpha$	**00**	$[0,4)$	-	$[1,4)$
	$1-\alpha \le p \le \alpha$	**0**	$[0,4)$	**1**	$[0,4)$
	$\alpha < p < 1$	-	$[0,3)$	**11**	$[0,4)$
$[0,3)$	$0 < p < 1/2$	**10**	$[0,4)$	**0**	$[0,4)$
	$1/2 \le p < 1$	**0**	$[0,4)$	**10**	$[0,4)$
$[1,4)$	$0 < p < 1/2$	**01**	$[0,4)$	**1**	$[0,4)$
	$1/2 \le p < 1$	**1**	$[0,4)$	**01**	$[0,4)$

Idea

Jeszcze Prościej

State	Prob{MPS}	LPS input Output	LPS input Next state	MPS input Output	MPS input Next state
$[0,4)$	$1/2 \le p \le \alpha$	**0**	$[0,4)$	**1**	$[0,4)$
	$\alpha < p < 1$	**00**	$[0,4)$	-	$[1,4)$
$[1,4)$	$1/2 \le p < 1$	**01**	$[0,4)$	**1**	$[0,4)$

More Probable Symbol występuje z prawdopodobieństwem większym niż 1/2. (Langdon i Rissanen)

Idea

Najprościej

	LPS input		MPS input	
State	Output	Next state	Output	Next state
$[0,4)$	**00**	$[0,4)$	-	$[1,4)$
$[1,4)$	**01**	$[0,4)$	**1**	$[0,4)$

Czy to jeszcze działa?

Idea

Najprościej

State	LPS input		MPS input	
	Output	Next state	Output	Next state
$[0,4)$	**00**	$[0,4)$	-	$[1,4)$
$[1,4)$	**01**	$[0,4)$	**1**	$[0,4)$

Czy to jeszcze działa?
Tak, na przykład dla kodowania unarnego.

Maksymalnie Niezbalansowane Podziały

	0 (LPS) input		**1** (MPS) input	
State	Output	Next state	Output	Next state
$[0,8)$	**000**	$[0,8)$	-	$[1,8)$
$[1,8)$	**001**	$[0,8)$	-	$[2,8)$
$[2,8)$	**010**	$[0,8)$	-	$[3,8)$
$[3,8)$	**011**	$[0,8)$	**1**	$[0,8)$

Kod Eliasa

	0 (LPS) input		**1** (MPS) input	
State	Output	Next state	Output	Next state
$[0,2)/2$	**0**	STOP	**1**	$[0,4)/4$
$[0,4)/4$	**00**	STOP	-	$[1,4)/4$
$[1,4)/4$	**01**	STOP	**1**	$[0,8)/8$
$[0,8)/8$	**000**	STOP	-	$[1,8)/8$
$[1,8)/8$	**001**	STOP	-	$[2,8)/8$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Plan

ϵ-podział

Dla każdej maksymalnej nadmiarowej długości kodu ϵ, możemy podzielić przestrzeń prawdopodobieństw tak, by zagwarantować, że nasze przybliżenie prawdopodobieństw nigdy nie doda więcej niż ϵ do kodu na jedno zdarzenie.
Wybieramy $P_0, P_1, \ldots$ (przypuszczalnie prawdziwe) oraz estymatory $Q_0, Q_1, \ldots$.
Pamiętamy oszacowanie na kod nadmiarowy E?

ϵ-podział

Algorytm

1. Niech $i := 0$ i $Q_0 := \frac{1}{2}$.
2. Znajdujemy $P_{i+1} > Q_i$ takie, że $E(P_{i+1}, Q_i) = \epsilon$. Będziemy używać Q_i jako estymator wszystkich prawdopodobieństw p takich, że $Q_i < p \leqslant P_{i+1}$.
3. Znajdź $Q_{i+1} > P_{i+1}$ takie, że $E(P_{i+1}, Q_{i+1}) = \epsilon$. Po obliczeniu P_{i+2} w kroku 2 następnej iteracji, będziemy używać Q_{i+1} jako estymator wszystkich prawdopodobieństw $P_{i+1} < p \leqslant P_{i+2}$.
4. Zwiększamy i i cofamy się do 2 kroku.

ρ-podział

Możemy chcieć ograniczyć błąd względny tak, by kod nigdy nie przekraczał optymalną długość więcej niż $1 + \rho$ razy. Możemy podejść do problemu podobnie jak w przypadku ϵ-podziałów, ale procedura się nigdy nie kończy. Można jednak uzyskać częściowe ρ-podziały.

Plan

Konstrukcja drzewa

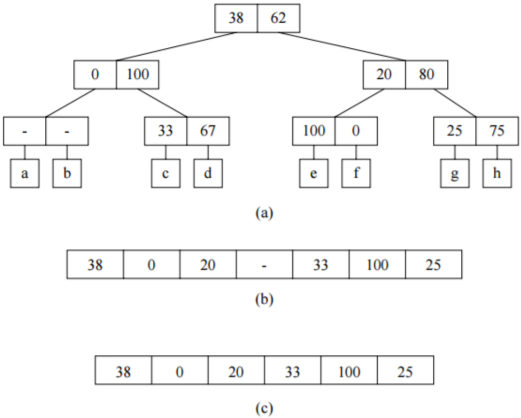
By użyć kodera zmniejszonej dokładności nad alfabetem wielosymbolowym, musimy stworzyć drzewo binarne.

- Drzewo Huffmana? Kosztowne.
- Liniowa reprezentacja drzewa? BFS.

Przykład Konstrukcji

Symbol	a	b	c	d	e	f	g	h
Prawdopodobieństwo	0	0	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{8}$	0	$\frac{1}{8}$	$\frac{3}{8}$

Konstrukcja drzewa



Rysunek: Konstrukcja drzewa skompresowanego. (a) Pełne drzewo binarne. (b) Reprezentacja liniowa. (c) Drzewo skompresowane.

Materiały

Paul G. Howard, Jeffrey Scott Vitter.
Practical Implementations of Arithmetic Coding