

Современная теория информации

Лекция 5. Кодирование дискретных источников при неизвестной статистике

Содержание лекции

- 1 Постановка задачи универсального кодирования;
- 2 Двухпроходное кодирование;
- 3 Нумерационное кодирование;
- 4 Адаптивное кодирование.

Универсальное кодирование

Постановка задачи

- Алгоритмы кодирования, такие как код Хаффмана или арифметическое кодирование предполагают знание распределения вероятностей символов источника.
- В реальных задачах данное распределение вероятностей **не известно**.
 - ▶ Можно оценить распределение вероятностей и передать его декодеру (двухпроходное кодирование или off-line coding).
 - ▶ Можно оценивать распределение вероятностей адаптивно одинаковым образом на стороне кодера и декодера используя уже закодированные/декодированные символы (адаптивное кодирование или online coding).

Универсальное кодирование

Постановка задачи

- Предположим, что мы будем кодировать *стационарные источники*.
- Для стационарного источника мы можем вычислить энтропию H , которая является нижней границей средней скорости кодирования R .
- "Хорошие" методы универсального кодирования должны обеспечить скорость кодирования, близкую к скорости кодирования, достигаемой в случае, если распределение вероятностей известно.

Универсальное кодирование

Постановка задачи

Предположим, что

- $\Omega = \{\omega\}$ – множество моделей источников.
- Если Ω – множество дискретных постоянных источников, то каждая ω определяется распределением вероятностей
 $\theta(\omega) = (\theta_1, \dots, \theta_K)$
- ω , а значит и θ , неизвестны как кодеру, так и декодеру.

Универсальное кодирование

Постановка задачи

Введем следующие обозначения:

- $H(\omega)$ энтропия для заданного ω .
- $\bar{R}_n(\omega)$ средняя по последовательности из n символов скорость кодирования для ω .
- $r_n(\omega) = \bar{R}_n(\omega) - H(\omega)$ — средняя избыточность для модели ω и длины последовательности n

Тогда, задача заключается в построении алгоритма, который минимизирует

$$r_n(\Omega) = \max_{\omega \in \Omega} r_n(\omega).$$

Если $\lim_{n \rightarrow \infty} r_n(\Omega) = 0$, то такое кодирование является **универсальным** для множества Ω

Универсальное кодирование

Постановка задачи

При разработке алгоритма универсального кодирования, необходимо учитывать (как для кодера, так и для декодера):

- Задержку. В соответствии с этим критерием, алгоритмы разделяются на:
 - ▶ Двухпроходные (**off-line**) алгоритмы.
 - ▶ Адаптивные (**on-line**) алгоритмы.
- Сложность.
- Объём памяти.

Двухпроходное кодирование

Общая идея

- Проход 1.
 - ① Оценить θ (обозначим оценку через $\hat{\theta}$).
 - ② Кодировать $\hat{\theta}$. На выходе получим кодовые слова c_1 .
- Проход 2.
 - ① Кодировать символы источника x используя $\hat{\theta}$. На выходе получим кодовые слова c_2 .
 - ② Сформировать кодовое слово из двух частей $c = (c_1, c_2)$.

Двухпроходное кодирование

На примере кода Хаффмана

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

$$I(\mathbf{x}) = I_1(\mathbf{x}) + I_2(\mathbf{x})$$

При равномерном кодировании получим $50 \times 8 = 400$ бит.

Двухпроходное кодирование

На примере кода Хаффмана

x	Число появлений x в x , $\tau(x)$	Длина кодового слова, $l(x)$	Кодовое слово	$\tau(x) \times l(x)$
I	1	6	010000	6
F	1	6	010001	6
_	12	2	00	24
W	5	3	100	15
E	4	4	0101	16
C	2	5	01001	10
A	4	4	1010	16
N	3	4	1011	12
O	5	3	110	15
T	1	6	011110	6
D	4	4	0110	16
S	3	4	1110	12
U	2	4	1111	8
L	2	5	01110	10
H	1	6	011111	6
Всего $l_2(x)$				178 бит

Двухпроходное кодирование

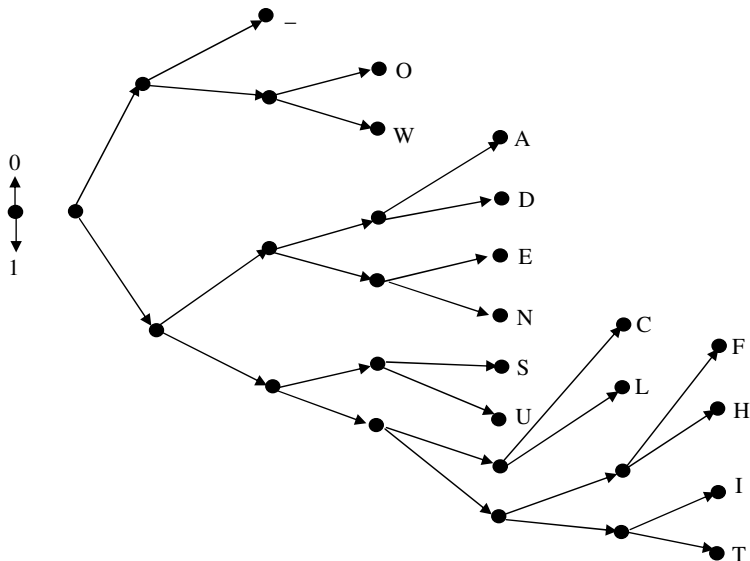
На примере канонического кода Хаффмана

- Для заданного распределения вероятностей можно построить несколько одинаково эффективных кодов Хаффмана.
- Код Хаффмана называется **каноническим**, если его короткие кодовые слова лексикографически предшествуют более длинным.

x	Длина кодового слова $l(x)$	Кодовое слово
_	2	00
O	3	010
W	3	011
A	4	1000
D	4	1001
E	4	1010
N	4	1011
S	4	1100
U	4	1101
C	5	11100
L	5	11101
F	6	111100
H	6	111101
I	6	111110
T	6	111111

Двухпроходное кодирование

На примере канонического кода Хаффмана



Двухпроходное кодирование

На примере канонического кода Хаффмана

- Достаточно указать количество концевых вершин для ярусов с номерами $0, \dots, l_{max}$, где l_{max} – максимальная длина кодового слова.

Ярус	Число вершин	Число концевых вершин n_i	Диапазон значений n_i	Затраты в битах
0	1	0	0...1	1
1	2	0	0...2	2
2	4	1	0...4	3
3	6	2	0...6	3
4	8	6	0...8	4
5	4	2	0...4	3
6	4	4	0...4	3
Total				19

$c_1 = (0\ 00\ 001\ 010\ 0110\ 010\ 100\ \text{ASCII}(x), \dots)$

$$l_1 = 19 + 8 \times 15 = 139 \text{ бит}, l = l_1 + l_2 = 139 + 178 = 317 \text{ бит}.$$

Двухпроходное кодирование

Скорость кодирования кодом Хаффмана

Theorem

Полное кодовое дерево, имеющее M конечных вершин, имеет $M - 1$ промежуточных вершин. Поэтому, $M + M - 1 = 2M - 1$ бит достаточно для описания полного описания дерева.

Двухпроходное кодирование

Скорость кодирования кодом Хаффмана

- $l_1(\mathbf{x}) \leq 2M - 1 + M \lceil \log M \rceil$
- $$l_2(\mathbf{x}) = \sum_{i=1}^n l(x_i) = \sum_{x \in X} \tau(x) l(x) = n \sum_{x \in X} \frac{\tau(x)}{n} l(x) =$$
$$n \sum_{x \in X} \hat{\theta}(x) l(x) = n E_{\hat{\theta}_n} \{l(x)\} \leq n(H(\hat{\theta}_n) + 1)$$
- $E(H(\hat{\theta}_n)) \leq H(E(\hat{\theta}_n)) = H(\theta) = H.$
- $$\bar{R}(\mathbf{x}) = \frac{l_1(\mathbf{x}) + l_2(\mathbf{x})}{n} \leq H + 1 + \frac{2M - 1 + M \lceil \log M \rceil}{n}$$

Нумерационное кодирование

Постановка задачи

- Последовательность на выходе источника $\mathbf{x} \in X^n$, $X = \{0, 1, \dots, M - 1\}$.
- Композиция $\boldsymbol{\tau}(\mathbf{x}) = (\tau_0(\mathbf{x}), \dots, \tau_{M-1}(\mathbf{x}))$, $M = |X|$.

Кодирование:

- Кодовое слово \mathbf{c} состоит из двух частей $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$.
- \mathbf{c}_1 описывает $\boldsymbol{\tau} = \boldsymbol{\tau}(\mathbf{x})$.
- \mathbf{c}_2 описывает номер \mathbf{x} в лексикографически упорядоченном списке всех возможных $\{\mathbf{x}\}$, которые имеют композицию $\boldsymbol{\tau}(\mathbf{x}) = \boldsymbol{\tau}$.

Нумерационное кодирование

Три способа кодирования композиции

- ❶ Кодируем каждый $\tau_i(\mathbf{x})$, кроме $i = M - 1$, прямым кодом, используя $\lceil \log(n + 1) \rceil$ бит.
- ❷ Представим композицию $\tau_0(\mathbf{x}), \dots, \tau_{M-1}(\mathbf{x})$ как двоичную последовательность вида $0^{\tau_0}10^{\tau_1}1, \dots, 10^{\tau_{M-1}}$, которая имеет длину $(n + M - 1)$ и вес (количество единиц) $M - 1$.
 - ▶ Количество строк такой же длины и веса: $N_\tau(n, M) = \binom{n+M-1}{M-1}$.
 - ▶ Лексикографически упорядочиваем все строки.
 - ▶ Кодируем равномерным кодом номер последовательности, используя $\lceil N_\tau(n, M) \rceil$ бит.
- ❸ Упорядочим Q ненулевых компонент $\boldsymbol{\tau}$ по убыванию, т.е., $\tau_0 \in \{1, \dots, n\}$, $\tau_1 \in \{1, \dots, \tau_0\}$, $\tau_2 \in \{1, \dots, \tau_1\}$ и т.д.
 - ▶ τ_Q кодируем АК с вероятностью $\frac{1}{n} \frac{1}{\tau_0} \frac{1}{\tau_1} \frac{1}{\tau_2} \dots \frac{1}{\tau_{Q-2}}$.
 - ▶ При помощи АК кодируем буквы, которые соответствуют компонентам композиции с вероятностью $\frac{1}{M} \frac{1}{M-1} \frac{1}{M-2} \dots \frac{1}{M-Q+2}$.

Нумерационное кодирование

Три способа кодирования композиции. Пример.

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

$n = 50, M = 256, Q = 15.$

① $l_1 = 255 \times \lceil \log 50 \rceil = 255 \cdot 6 = 1530 \text{ бит.}$

② $l_1 = \left\lceil \log \binom{255+50}{255} \right\rceil = 193 \text{ бит.}$

③ $\tau(\mathbf{x}) = (12, 5, 5, 4, 4, 4, 3, 3, 2, 2, 2, 1, 1, 1, 1, 0, \dots, 0)$

$$\begin{aligned} l_1 &= \left\lceil \log (50 \cdot 12 \cdot 5^2 \cdot 4^3 \cdot 3^2 \cdot 2^3 \times 256 \cdot 255 \dots \cdot 243) \right\rceil \\ &= 27 + 120 = 147 \text{ бит.} \end{aligned}$$

Нумерационное кодирование

Кодирование x при известном $\tau(x)$

Пусть $M = 3$, и $\tau = (\tau_0, \tau_1, \tau_2)$. Тогда количество всех возможных x для $\tau(x)$

$$N(\tau) = \binom{n}{\tau_0} \binom{n - \tau_0}{\tau_1} = \frac{n!}{\tau_0!(n - \tau_0)!} \frac{(n - \tau_0)!}{\tau_1!(n - \tau_0 - \tau_1)!} = \frac{n!}{\tau_0!\tau_1!\tau_2!}$$

В общем случае, для алфавита объёмом M имеем:

$$N(\tau) = \frac{n!}{\tau_0!\tau_1!\dots\tau_{M-1}!}$$

Нумерационное кодирование

Кодирование x при известном $\tau(x)$. Пример.

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

$$l_2 = \left\lceil \log \frac{50!}{12! (5!)^2 (4!)^3 (3!)^2 (2!)^3} \right\rceil = 150 \text{ бит.}$$

$$l = l_1 + l_2 = 147 + 150 = 297 \text{ бит.}$$

Нумерационное кодирование

Кодирование x при известном $\tau(x)$ арифметическим кодером. Пример.

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

t	x	$\hat{p}(x)$	Композиция $\tau(x)$
0	—	—	12,5,5,4,4,4,3,3,2,2,2,1,1,1,1
1	I	1/50	12,5,5,4,4,4,3,3,2,2,2,1,1,1,0
2	F	1/49	12,5,5,4,4,4,3,3,2,2,2,1,1,0
3	_	12/48	11,5,5,4,4,4,3,3,2,2,2,1,1
4	W	5/47	11,4,5,5,4,4,4,3,3,2,2,2,1,1
5	E	4/46	11,4,5,5,3,4,4,3,3,2,2,2,1,1
6	_	10/45	10,4,5,5,3,4,4,3,3,2,2,2,1,1
...

$$G = \frac{12!(5!)^2(4!)^3(3!)^2(2!)^3}{50!}$$

$$L = \lceil -\log G \rceil + 1 = 151 \text{ бит.}$$

Нумерационное кодирование

Теоретический анализ избыточности

Аппроксимация Стирлинга:

$$n! \approx \sqrt{2\pi n} n^n e^{-n}$$

$$\log N(\tau) \approx nH\left(\frac{\tau}{n}\right) - \frac{M-1}{2}\log(2\pi n)$$

$$\log N_\tau(n, M) \approx (M-1)\log(n+1)$$

Поэтому,

$$\bar{R} \approx H + \frac{M-1}{2} \frac{\log n}{n} + \frac{\text{const}}{n} \xrightarrow{n \rightarrow \infty} H$$

Это минимальная достижимая избыточность.

Адаптивное кодирование

Общая идея

- Кодеру не доступны сообщения, которые появятся в будущем, т.е., при кодировании x_i , сообщения x_{i+1}, x_{i+2}, \dots считаются неизвестными.
- По последовательности уже закодированных сообщений x_0, x_1, \dots, x_{i-1} кодер оценивает вероятность для символа x_i и строит для него код в соответствии с этой оценкой.
- После декодирования сообщений x_0, x_1, \dots, x_{i-1} декодер оценивает вероятность для символа x_i так же как и кодер, после чего декодирует x_i .

Адаптивное кодирование

Оценка вероятности с использованием счётчиков

Пусть необходимо передать $\mathbf{x} = (x_1, \dots, x_n)$ арифметическим кодером. Для этого каждому символу x_t необходимо сопоставить $\hat{p}_t(a)$ – оценку вероятности того, что $x_t = a$, $a = 1, \dots, M$. Предположим, что x_1, \dots, x_{t-1} уже переданы и известны декодеру. Тогда

$$\hat{p}_t(a) = \frac{\tau_t(a)}{t}, \text{ где } \tau_t(a) \text{ число символов } a \text{ в } x_1, \dots, x_{t-1}.$$

$$\hat{p}_t(a) = \frac{\tau_t(a) + 1}{t + M}, \text{ поправка, чтобы избежать нулевых вероятностей.}$$

$$\hat{p}_t(a) = \frac{\tau_t(a) + 1/2}{t + M/2}.$$

Адаптивное кодирование

Оценка вероятности с использованием счётчиков

Пусть необходимо передать $\mathbf{x} = (x_1, \dots, x_n)$ арифметическим кодером. Для этого каждому символу x_t необходимо сопоставить $\hat{p}_t(a)$ – оценку вероятности того, что $x_t = a$, $a = 1, \dots, M$. Предположим, что x_1, \dots, x_{t-1} уже переданы и известны декодеру. Тогда

$$\hat{p}_t(a) = \frac{\tau_t(a)}{t}, \text{ где } \tau_t(a) \text{ число символов } a \text{ в } x_1, \dots, x_{t-1}.$$

$$\hat{p}_t(a) = \frac{\tau_t(a) + 1}{t + M}, \text{ поправка, чтобы избежать нулевых вероятностей.}$$

$$\hat{p}_t(a) = \frac{\tau_t(a) + 1/2}{t + M/2}.$$

Адаптивное кодирование

Оценка вероятности с использованием счётчиков

Пусть необходимо передать $\mathbf{x} = (x_1, \dots, x_n)$ арифметическим кодером. Для этого каждому символу x_t необходимо сопоставить $\hat{p}_t(a)$ – оценку вероятности того, что $x_t = a$, $a = 1, \dots, M$. Предположим, что x_1, \dots, x_{t-1} уже переданы и известны декодеру. Тогда

$$\hat{p}_t(a) = \frac{\tau_t(a)}{t}, \text{ где } \tau_t(a) \text{ число символов } a \text{ в } x_1, \dots, x_{t-1}.$$

$$\hat{p}_t(a) = \frac{\tau_t(a) + 1}{t + M}, \text{ поправка, чтобы избежать нулевых вероятностей.}$$

$$\hat{p}_t(a) = \frac{\tau_t(a) + 1/2}{t + M/2}.$$

Адаптивное кодирование

Оценка вероятности с использованием счётчиков

Пусть необходимо передать $\mathbf{x} = (x_1, \dots, x_n)$ арифметическим кодером. Для этого каждому символу x_t необходимо сопоставить $\hat{p}_t(a)$ – оценку вероятности того, что $x_t = a$, $a = 1, \dots, M$. Предположим, что x_1, \dots, x_{t-1} уже переданы и известны декодеру. Тогда

$$\hat{p}_t(a) = \frac{\tau_t(a)}{t}, \text{ где } \tau_t(a) \text{ число символов } a \text{ в } x_1, \dots, x_{t-1}.$$

$$\hat{p}_t(a) = \frac{\tau_t(a) + 1}{t + M}, \text{ поправка, чтобы избежать нулевых вероятностей.}$$

$$\hat{p}_t(a) = \frac{\tau_t(a) + 1/2}{t + M/2}.$$

Адаптивное кодирование

Оценка вероятности с использованием счётчиков. Пример.

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

$$\tau = (12, 5, 5, 4, 4, 4, 3, 3, 2, 2, 2, 1, 1, 1, 1, 0, \dots)$$

t	x	$\hat{p}_t(a) = \frac{\tau_t(a)+1}{t+M}$	$\hat{p}_t(a) = \frac{\tau_t(a)+1/2}{t+M/2}$
0	I	$(0+1)/(0+256)=1/256$	$(2*0+1)/(2*0+256)=1/256$
1	F	$(0+1)/(1+256)=1/257$	$(2*0+1)/(2*1+256)=1/258$
2	—	$(0+1)/(2+256)=1/258$	$(2*0+1)/(2*2+256)=1/260$
3	W	$(0+1)/(3+256)=1/259$	$(2*0+1)/(2*3+256)=1/262$
4	E	$(0+1)/(4+256)=1/260$	$(2*0+1)/(2*4+256)=1/264$
5	—	$(1+1)/(5+256)=2/261$	$(2*1+1)/(2*5+256)=3/266$
...
12	—	$(2+1)/(12+256)=3/268$	$(2*2+1)/(2*12+256)=5/280$
...
15	—	$(3+1)/(15+256)=4/271$	$(2*3+1)/(2*15+256)=7/286$
...
49	—	$(2+1)/(49+256)=3/305$	$(2*2+1)/(2*49+256)=5/354$

Адаптивное кодирование

Оценка вероятности с использованием счётчиков. Пример.

$$\textcircled{1} \quad \hat{p}_t(a) = \frac{\tau_t(a)+1}{t+M}$$

$$\triangleright G = 1 \cdot \frac{1}{256} \cdot \frac{1}{257} \cdot \frac{1}{258} \cdot \frac{1}{259} \frac{1}{260} \cdot \frac{2}{261} \cdots = \frac{12!(5!)^2(4!)^3(3!)^2(2!)^3}{256 \cdot 257 \cdots 305}.$$

$$\triangleright L = \lceil -\log G \rceil + 1 = 343 \text{ бит.}$$

$$\textcircled{2} \quad \hat{p}_t(a) = \frac{\tau_t(a)+1/2}{t+M/2}$$

$$\triangleright G = 1 \cdot \frac{1}{256} \cdot \frac{1}{258} \cdot \frac{1}{260} \cdot \frac{1}{262} \cdot \frac{1}{264} \cdot \frac{3}{266} \cdots = \frac{(23)!!(9!!)^2(7!!)^3(5!!)^2(3!!)^3}{256 \cdot 258 \cdots 354}.$$

$$\triangleright L = \lceil -\log G \rceil + 1 = 323 \text{ бит.}$$

Адаптивное кодирование

Алгоритмы A и D

- Можно использовать подход основанный на так называемом esc-символе.
- В этом случае, мы добавляем дополнительный символ в алфавит. Этот символ передаётся, если на вход приходит символ, который ранее не появлялся.

Адаптивное кодирование

Алгоритмы A и D

Общая идея:

- Используется оценка $p_t(a) = \frac{\tau_t(a)}{t+1}$, если $\tau_t(a) > 0$.
- Передаётся “esc”, если $\tau_t(a) = 0$, $p_t(esc) = \frac{1}{t+1}$

Алгоритм A:

$$\hat{p}_t(a) = \begin{cases} \frac{\tau_t(a)}{t+1}, & \text{если } \tau_t(a) > 0; \\ \frac{1}{t+1} \frac{1}{M-M_t}, & \text{если } \tau_t(a) = 0, \end{cases}$$

M_t – число различных символов, встретившихся на момент времени t .

Адаптивное кодирование

Алгоритмы A и D

Алгоритм D:

$$\hat{p}_t(a) = \begin{cases} \frac{\tau_t(a) - 1/2}{t}, & \text{если } \tau_t(a) > 0; \\ \frac{1}{M}, & \text{если } \tau_t(a) = 0, t = 0 \\ \frac{M_t}{2t} \frac{1}{M - M_t}, & \text{если } \tau_t(a) = 0, t > 0 \end{cases}$$

Адаптивное кодирование

Алгоритмы A и D. Пример.

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

$$\tau = (12, 5, 5, 4, 4, 4, 3, 3, 2, 2, 2, 1, 1, 1, 1, 0, \dots)$$

t	x	\hat{p}_A	\hat{p}_D
0	I	$1 \cdot 1/256$	$1/256$
1	F	$1/2 \cdot 1/255$	$1/2 \cdot 1/255$
2	—	$1/3 \cdot 1/254$	$2/4 \cdot 1/254$
3	W	$1/4 \cdot 1/253$	$3/6 \cdot 1/253$
4	E	$1/5 \cdot 1/252$	$4/8 \cdot 1/252$
5	—	$1/6$	$1/10$
...
12	—	$2/13$	$3/24$
...
15	—	$3/16$	$5/30$
...
49	N	$2/50$	$3/98$

Адаптивное кодирование

Алгоритмы A и D. Пример.

$$G_A = \frac{11!(4!)^2(3!)^3(2!)^2(1!)^3}{50!} \cdot \frac{1}{256 \cdot 255 \cdot \dots \cdot 242}$$

$$L_A = \lceil -\log G_A \rceil + 1 = 291 \text{ bits}$$

$$G_D = \frac{(2 \cdot 12 - 3)!!((2 \cdot 5 - 3)!!)^2((2 \cdot 4 - 3)!!)^3((2 \cdot 3 - 3)!!)^2}{98!!}$$

$$\times \frac{14!}{256 \cdot 255 \cdot \dots \cdot 242}$$

$$L_D = \lceil -\log G_2 \rceil + 1 = 287 \text{ бит},$$

где

$$n!! = \begin{cases} 1 \cdot 3 \dots \cdot n, & n - \text{нечетные.} \\ 2 \cdot 4 \dots \cdot n, & n - \text{четные.} \end{cases}$$

Адаптивное кодирование

Избыточность адаптивного арифметического кодирования

Theorem

При кодировании дискретного постоянного источника с энтропией H , средняя скорость адаптивного арифметического кодирования удовлетворяет неравенству

$$\bar{R} \leq H + \frac{M \log(n+1) + K}{2n},$$

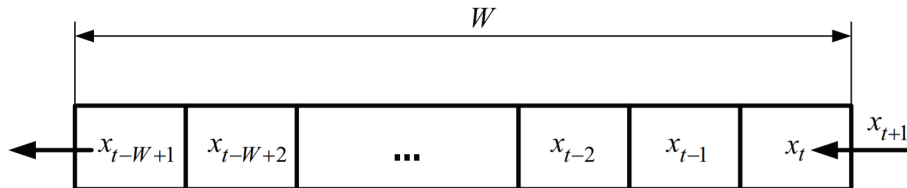
где K не зависит от длины последовательности n .

Сравнение алгоритмов

Алгоритм	Количество проходов	Битовые затраты
Двухпроходное кодирование кодом Хаффмана	2	317
Нумерационное кодирование	2	298
Адаптивное арифметическое кодирование, \hat{p}_A	1	291
Адаптивное арифметическое кодирование, \hat{p}_D	1	287
7z	1	289-296

Адаптивное арифметическое кодирование

Оценка вероятности при помощи "скользящего окна"



$$\hat{p}_t(a) = \frac{\tau_t(a) + 1}{W + 1}.$$

Адаптивное арифметическое кодирование

Аппроксимация "скользящего окна" для двоичного случая

- Обозначим через s_t количество единиц в скользящем окне после кодирования t символов.
- Можно не хранить окно в памяти, а s_t аппроксимировать следующим образом:
 - 1 Из окна удаляется среднее число единиц:

$$s_{t+1} \leftarrow s_t - \frac{s_t}{W}.$$

- 2 Новый символ добавляется в окно:

$$s_{t+1} \leftarrow s_{t+1} + x_t.$$

Итоговое правило обновления:

$$s_{t+1} = \left(1 - \frac{1}{W}\right) \cdot s_t + x_t.$$

Адаптивное арифметическое кодирование

Оценка вероятности в M-coder

$$\begin{cases} \hat{p}_s = (1 - \gamma)\hat{p}_{s-1}, \text{ where } s = 1, \dots, 63, \hat{p}_0 = 0.5, \\ \gamma = 1 - \left(\frac{\hat{p}_{min}}{0.5}\right)^{\frac{1}{63}}, \hat{p}_{min} = 0.01875. \end{cases}$$

Оценка вероятности для символа x_{t+1} вычисляется как:

$$\hat{p}_{t+1} = \begin{cases} (1 - \gamma)\hat{p}_t + \gamma, & \text{if } x_t = \text{LPS}, \\ \max\{(1 - \gamma)\hat{p}_t, \hat{p}_{min}\}, & \text{if } x_t = \text{MPS}, \end{cases}$$

Адаптивное арифметическое кодирование

Реализация адаптивного двоичного арифметического кодирования без умножений

```
1:  $T \leftarrow R \times p(x_t)$   
2:  $R \leftarrow R - T$   
3: if  $x_t = 1$  then  
4:    $L \leftarrow L + R$   
5:    $R \leftarrow T$   
6: end if  
7: call Ренормализация
```

```
1:  $T \leftarrow R \times p(x_t)$   
2:  $R \leftarrow R - T$   
3: if  $F < R$  then  
4:    $x_t = 0$   
5: else  
6:    $L \leftarrow L + R$   
7:    $R \leftarrow T$   
8:    $x_t = 1$   
9: end if  
10: call Ренормализация
```

Адаптивное арифметическое кодирование

Реализация адаптивного двоичного арифметического кодирования без умножений

```
1: while  $R < 2^{b-2}$  do
2:   if  $L \geq 2^{b-1}$  then
3:     WriteOnes(1)
4:     WriteZeros(bits_to_follow), bits_to_follow  $\leftarrow 0$ 
5:      $L \leftarrow L - 2^{b-1}$ 
6:   else if  $L < 2^{b-2}$  then
7:     WriteZeros(1)
8:     WriteOnes(bits_to_follow), bits_to_follow  $\leftarrow 0$ 
9:   else
10:    bits_to_follow  $\leftarrow \textit{bits\_to\_follow} + 1$ 
11:     $L \leftarrow L - 2^{b-2}$ 
12:   end if
13:    $L \leftarrow L \ll 1, R \leftarrow R \ll 1$ 
14: end while
```

Адаптивное арифметическое кодирование

Реализация адаптивного двоичного арифметического кодирования без умножений

После ренормализации регистр R находится в интервале:

$$\frac{1}{2}2^{b-1} \leq R < 2^{b-1}.$$

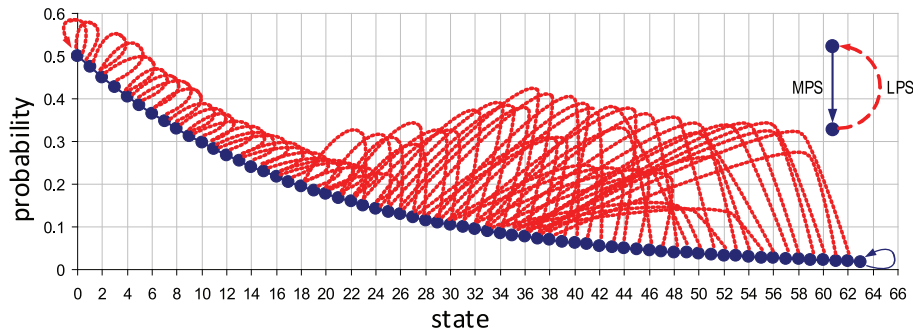
Поэтому умножением может быть аппроксимировано следующим образом:

$$T = R \times \hat{p}_t \approx \alpha 2^{b-1} \times \hat{p}_t,$$

где $\alpha \in [\frac{1}{2}, \dots, 1)$. Для улучшения точности M-coder квантует интервал $[\frac{1}{2}2^{b-1}; 2^{b-1})$ равномерно на 4 интервала. Затем, для каждого из четырёх интервалов результат умножения $R \times \hat{p}_s$ помещается в таблицу $TabRangeLPS[s][\Delta]$, состоящую из 64×4 значений.

Адаптивное арифметическое кодирование

Реализация адаптивного двоичного арифметического кодирования без умножений



Двоичное арифметическое кодирование

Реализация M-coder¹

```
1:  $\Delta \leftarrow (R - 2^{b-2}) \gg (b - 4)$ 
2:  $T \leftarrow \text{TabRangeLPS}[s][\Delta]$ 
3:  $R \leftarrow R - T$ 
4: if  $x_i \neq \text{MPS}$  then
5:    $L \leftarrow L + R$ 
6:    $R \leftarrow T$ 
7:   if  $s = 0$  then
8:      $\text{MPS} \leftarrow !\text{MPS};$ 
9:   end if
10:   $s \leftarrow \text{TransStateLPS}[s]$ 
11: else
12:   $s \leftarrow \text{TransStateMPS}[s]$ 
13: end if
14: call Renormalization procedure
```

¹D. Marpe, T. Wiegand, "A Highly Efficient Multiplication-Free Binary Arithmetic Coder and Its Application in Video Coding," IEEE ICIP, 2003.

Адаптивное арифметическое кодирование

Реализация без умножений и таблиц²

Умножим обе части правила на $\alpha 2^{b-1}$:

$$s'_{t+1} = \left(1 - \frac{1}{W}\right) \cdot s'_t + \alpha 2^{b-1} x_t,$$

где $s'_t = \alpha 2^{b-1} s_t$. Целочисленная реализация:

$$s'_{t+1} = \begin{cases} s'_t + \left\lfloor \frac{\alpha 2^{b-1} 2^w - s'_t + 2^{w-1}}{2^w} \right\rfloor, & \text{if } x_t = 1 \\ s'_t - \left\lfloor \frac{s'_t + 2^{w-1}}{2^w} \right\rfloor, & \text{if } x_t = 0, \end{cases}$$

Тогда умножение может быть аппроксимировано следующим образом:

$$T = R \times \hat{p}_t \approx \alpha 2^{b-1} \times \hat{p}_t = \frac{s'_t}{2^w}.$$

²E.Belyaev, A.Turlikov, K.Egiazarian and M.Gabbouj, An efficient adaptive binary arithmetic coder with low memory requirement // IEEE Journal of Selected Topics in Signal Processing. Special Issue on Video Coding: HEVC and beyond, 2013.

Адаптивное арифметическое кодирование

Реализация без умножений и таблиц

Для увеличения точности аппроксимации интервал $[\frac{1}{2}2^{b-1}; 2^{b-1})$ квантуется на 4 подинтервала:

$$\left\{ \frac{9}{16}2^{b-1}, \frac{11}{16}2^{b-1}, \frac{13}{16}2^{b-1}, \frac{15}{16}2^{b-1} \right\}.$$

Для этого сначала вычисляется s'_t для $\alpha = \frac{9}{16}$. Затем, умножение аппроксимируется как:

$$T = R \times \hat{p}_t \approx \frac{s'_t + \Delta \times \frac{1}{4}s'_t}{2^w}, \text{ where } \Delta = \frac{R - 2^{b-2}}{2^{b-4}}.$$

Адаптивное арифметическое кодирование

Реализация без умножений и таблиц

```
1:  $\Delta \leftarrow (R - 2^{b-2}) \gg (b - 4)$ 
2:  $T \leftarrow (s + \Delta \times (s \gg 2)) \gg w$ 
3:  $T \leftarrow \max(1, T)$ 
4:  $R \leftarrow R - T$ 
5: if  $x_i \neq \text{MPS}$  then
6:    $L \leftarrow L + R$ 
7:    $R \leftarrow T$ 
8:    $s \leftarrow s + ((\alpha 2^{b-1} 2^w - s + 2^{w-1}) \gg w)$ 
9:   if  $s > \alpha 2^{b-2} 2^w$  then
10:     $\text{MPS} \leftarrow !\text{MPS};$ 
11:     $s \leftarrow \alpha 2^{b-2} 2^w;$ 
12:   end if
13: else
14:    $s \leftarrow s - ((s + 2^{w-1}) \gg w)$ 
15: end if
16: call Renormalization procedure
```