Арифметическое кодирование

В основе алгоритма арифметического кодирования лежит очень простая и красивая идея. Входной поток байтов рассматривается как запись некоторого числа, в которой входные символы играют роль "цифр". Алгоритм переводит эту запись в двоичное представление следующим образом.

Пусть мы имеем входную последовательность длиной n байт. Обозначим частоту появления байта со значением k в этой входной последовательности через α_k, $k = 0, \ldots, 255$. Здесь под частотой мы понимаем отношение количества появлений данного фиксированного байта к количеству всех байт. Таким образом, $0 \leq \alpha_k \leq 1$, $\sum_{k=0}^{255} \alpha_k = 1$, и количество появлений данного байта есть $p_k = \alpha_k n$.

Разобъем отрезок $s_0 = [0, 1]$ на подотрезки, равные по длине α_k и сопоставим эти отрезки соответствующим значениям байтов входной последовательности.

Возьмем первый байт входного потока и рассмотрим соответствующий ему отрезок (обозначим его s_1). Разобъем отрезок s_1 пропорционально частотам α_k (аналогично тому, как был разбит отрезок s_0). Теперь возьмем следующий входной байт и рассмотрим соответствующий ему отрезок $s_2 \subset s_1$. Далее процесс продолжается аналогично (т.е. разбиваем текущий выбранный отрезок пропорционально частотам и выбираем в нем следующий отрезок по новому поступившему байту входной последовательности). В результате мы получим последовательность вложенных друг в друга отрезков $s_0 \supset s_1 \supset s_2 \supset \ldots \supset s_n$. Результатом кодирования является любое число, принадлежащее последнему выбранному отрезку s_n. Декодирование производится достаточно просто. Для этого надо взять кодирующее число и проверить какому из отрезков разбиения оно принадлежит. Эта проверка даст первый байт исходного набора данных. Затем можно определить какому второму отрезку принадлежит число и т.д. Для устранения неоднозначности кодирования и декодирования следует исключить из проверки одну из границ отрезков (например, правую). Очевидно также, что для возможности декодирования следует знать таблицу частот и общее количество байтов в исходной последовательности (иначе неясно в какой момент прекращать процесс декодирования).

Опишем алгоритм построения кодирующей дроби в виде псевдокода. Пусть элементы массива `a[i]` задают точки границ отрезков частотного разбиения для байта, т.е. $\alpha_i = \mathtt{a[i+1]} - \mathtt{a[i]}$, а функция `NextByte()` читает и возвращает очередной байт входной последовательности, либо значение `EOI` при окончании набора данных (EOI — End Of Information). Описанный выше алгоритм выглядит так:

```
left = 0;
right = 1;
while ( (k=NextByte()) != EOI ) {
   d = right - left;
   right = left + d*a[k+1];
   left  = left + d*a[k];
}
```

По окончании этого цикла в качестве кодирующей дроби можно взять любое число `x`, удовлетворяющее условию $\mathtt{left} \leq \mathtt{x} < \mathtt{right}$.

Декодирование производится обращением описанного выше процесса.

```
x = <кодирующая дробь>
```

```
n = <общее число символов в исходном файле>
while (n--) {
  k = Interval (x);     // найти номер интервала, содержащего x
  Output(Symbol(k));    // вывести k-й символ
  d = a[k+1]-a[k]       // длина интервала для k-го символа
  x = (x - a[k])/d;     // преобразуем дробь для следующего символа
}
```

Алгоритм арифметического кодирования может привести к поразительным результатам. Например, если при кодировании окажется, что окончательный отрезок содержит точку $0.5_{10} = 0.1_2$, то мы получим выходной код длиной в 1 бит! (не считая, конечно, накладных расходов в виде таблицы частот и количества входных байтов). Однако, в общем случае на это нельзя полагаться, и возникает резонный вопрос: а почему этот алгоритм в принципе способен сжимать информационно избыточные наборы данных?

Теорема. *Для заданной входной последовательности длины N символов метод арифметического кодирования дает код, длина которого не больше $L \approx NE$, где E — информационная энтропия таблицы частот символов входной последовательности.*

(информационная энтропия — количество информации по Шеннону $= \sum \alpha_i \log \alpha_i^{-1}$).

Доказательство. Пусть α_k — длины отрезков исходного разбиения и $p_k = \alpha_k N$ — количество появлений байта со значением k. Нетрудно увидеть, что длина последнего кодирущего отрезка равна

$$l = \prod_{k:\alpha_k \neq 0} \alpha_k^{p_k}.$$

Для построения кода нам нужно найти число, принадлежащее такому отрезку и имеющее по возможности наименьшее количество бит в своем двоичном представлении. Для того, чтобы в произвольном отрезке длины l нашлась двоичная дробь, содержащая L значащих разрядов, достаточно выполнения неравенства $2^{-L} \leq l$. Отсюда получаем $L \geq -\log_2 l$. Так как нас интересует дробь с наименьшим количеством значащих разрядов, то принимаем $L \approx -\log_2 l$ за длину выходного кода алгоритма, т.е.

$$L \approx -\log_2 \prod_{k:\alpha_k \neq 0} \alpha_k^{p_k} = \sum_{k:\alpha_k \neq 0} N\alpha_k \log_2 \alpha_k = NE.$$

Следствие. *Пусть входная последовательность, состоящая из N байт (или $8N$ бит), имеет неравномерное распределение частот значений отдельных байтов. Тогда кодирующая дробь метода арифметического кодирования может быть представлена менее чем $8n$ битами.*

Лемма. *Пусть функция $f(x)$ является выпуклой вниз на некотором интервале (a, b). Тогда для любого набора $x_1, \ldots, x_m \in (a, b)$ выполнено соотношение*

$$\frac{1}{m}\sum_{k=1}^{m} f(x_k) \geq f\Big(\frac{1}{m}\sum_{k=1}^{m} x_k\Big),$$

причем равенство достигается только при $x_1 = x_2 = \ldots = x_m$.

Доказательство легко проводится индукцией по m с учетом хорошо известного неравенства для выпуклых вниз функций: $\alpha f(x_1) + (1-\alpha) f(x_2) > f(\alpha x_1 + (1-\alpha)x_2)$, $0 < \alpha < 1$.

Доказательство следствия. Для байтовой последовательности таблица частот $\{\alpha_0, \ldots, \alpha_{255}\}$.

$$E = -256N \frac{1}{256} \sum_{k:\alpha_k \neq 0} \alpha_k \log_2 \alpha_k \leq -N \cdot 256 \Big(\frac{\alpha_0 + \ldots + \alpha_{255}}{256} \Big) \log_2 \Big(\frac{\alpha_0 + \ldots + \alpha_{255}}{256} \Big) =$$

$$= -N \log_2 \frac{1}{256} = 8N.$$

Так как не все α_k одинаковые, то имеем знак $<$ в неравенстве.

Алгоритм арифметического кодирования имеет и адаптивный вариант, который по принципам построения аналогичен адаптивному методу Хаффмена. Этот алгоритм можно описать следующим образом (на примере байтовой последовательности)

1. Строится таблица частот дла равномерного распределения (т.е. все отрезки разбиения имеют длину $1/256$).

2. Очередной отрезок разбиения выбирается по очередному входному символу на основании текущей таблицы частот.

3. Таблица частот пересчитывается с учетом появления только что обработанного входного символа.

4. Продолжаем обработку по пунктам 2–4, пока есть входные символы.

5. В окончательном отрезке разбиения выбираем кодирующую дробь.

Адаптивное декодирование строится аналогично.

1. Строится таблица частот дла равномерного распределения (т.е. все отрезки разбиения имеют длину $1/256$).

2. Ищется очередной отрезок, содержащий кодирующую дробь, на основании текущей таблицы частот. Выходной символ определяется по найденному отрезку.

3. Таблица частот пересчитывается с учетом появления только что полученного выходного символа.

4. Продолжаем обработку по пунктам 2–4, пока есть не исчерпается общее количество символов.