Блоковое кодирование. Прямая и обратная теоремы кодирования

5 января 2023 г. 1:22

Код Гилберта-Мура лучше кода Шеннона отсутствием необходимости сортировать вероятности, но хуже с точки зрения избыточности. Применять такой код при побуквенном кодировании не имеет смысла как раз из-за большой избыточности. Чтобы уменьшить кодовую избыточность, можно использовать блоковое кодирование. Чем больше длина блока, тем меньше избыточность на символ ($R = \bar{l} - H(X^n) = \bar{l} - nH_n(X)$)

1. Пусть $x \in X = \{0, 1, \dots, M-1\}$. Последовательность на выходе источника будем кодировать блоками $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$.
2. Каждый блок $\boldsymbol{x}$ длины n может рассматриваться как буква нового укрупнённого алфавита из всех комбинаций векторов длины n.
3. Применим любой алгорим побуквенного кодирования для укрупнённого алфавита.

Энтропия на блок для укрупненного алфавита вычисляется следующим образом:

x - буква укрупненного алфавита

Энтропия укрупнённого алфавита:

$$H(X^n) = -\sum_{\boldsymbol{x} \in X^n} p(\boldsymbol{x}) \log_2 p(\boldsymbol{x}).$$

Далее применим побуквенное кодирование к блокам

вообще формула $\bar{l} \leq H + r_n$, а тут $=$, т.к. в общем виде

Пусть r_n – избыточность укрупнённого алфавита, тогда $\bar{l} = H(X^n) + r_n$. Средние затраты на символ исходного алфавита:

избыточность Хаффман=1 Шеннон=1 Гилберт-Мур=2

$$\bar{R} = \frac{H(X^n) + r_n}{n} = \frac{H(X^n)}{n} + \frac{r_n}{n}.$$

скорость кода, показывает, сколько бит мы тратим на символ

Если устремить n к бесконечности в формуле

то $\frac{r_n}{n}$ будет стремиться к уменьшению

а $\frac{H(X^n)}{n}$ - ни что иное, как энтропия на символ (в данном случае на блок, т.к. каждый символ нового алфавита - по сути блок)

- Для источника без памяти $H(X^n) = nH(X)$ получим:

$$\bar{R} = H(X) + \frac{r_n}{n}.$$

Если $n \to \infty$, то $\frac{r_n}{n} \to 0$.

- Для источника с памятью $H(X^n) \leq nH(X)$, и поэтому $\frac{H(X^n)}{n} \leq H(X)$

$$\lim_{n \to \infty} \frac{H(X^n)}{n} = H_\infty(X)$$

Прямая и обратная теоремы кодирования

Обратная теорема кодирования

$\overline{R} \geq H_\infty(X)$ (средняя длина на символ >= энтропии, которая вычисляется при устремлении размера блока к бесконечности)

Т.к. для кода Шеннона $\bar{l} \leq H+1$, где 1 - это r, т.е. избыточность

Прямая теорема кодирования

Можно найти такой префиксный код, для которого будет верно неравенство $\overline{R} \leq H_\infty(X) + \epsilon$, где ϵ - какое-то небольшое положительное число. То есть мы устремляем n к ∞ и можем достигнуть этого ϵ.

Доказательство

1. Берем исходный алфавит и укрупняем его. Получаем новый алфавит, в котором каждая буква - блок длиной n символов. Понятно, что средняя длина кодового слова будет больше или равна энтропии блока из n символов
   Т.е.: $\boldsymbol{x} \in X^n$, $\overline{l}_n \geq H(X^n)$
2. Для любого n существует код (например, Шеннона), такой что $\overline{l}_n \leq H(X^n) + 1$
3. Вычисляем среднюю длину на символ, получаем, что она больше или равна энтропии при устремлении n к ∞: $R_n = \frac{\overline{l}_n}{n} \geq \frac{H(X^n)}{n} \geq H_\infty(X)$ для всех n
   $\geq$, т.к. $\bar{l}_n \geq H(X^n)$
   (обратная теорема доказана)
4. Если мы в обратную сторону посмотрим, то можно найти такое большое n, что избыточность, которую мы прибавляем, будет стремиться к значению $H_\infty(X) + \epsilon$
   Т.е.: для $n \geq n_0$ и $\epsilon > 0$ $R_n \leq H_\infty(X) + \epsilon$

для n=2 реально реализовать, а для n=10, например, памяти не хватит (новый алфавит не влезет в переменную?)

Этот подход позволяет сжать файл близко к энтропии

1. Код Хаффмана для укрупнённого алфавита сложно использовать на практике. Если $|X| = 256$, то для $n = 2$ $|X^n| = 65536$.
   большие алфавиты проблематично сортировать
2. Код Шеннона сложно использовать, так как он требует сортировки.
   реализуется, но большая избыточность

3. Код Гилберта-Мура хорошо подходит для кодирования блоков.
4. Арифметическое кодирование является обобщением кода Гилберта-Мура для случая блокового кодирования.