

1 Введение

Конспект лекций и практики по Теории информации.

В формулах матрицы обозначаются прописной буквой. Строчной буквой обозначаются скаляры. Если строчная буква выделена жирным, то это вектор. Бывают исключения.

2 Измерение информации. Энтропия

2.1 Общая схема системы связи

Общая схема системы связи:

```
source -> source -> channel -> channel -> channel -> source -> user  
        encoder      encoder      decoder      decoder
```

Источник информации порождает сообщение. Задача кодировщика представить сообщение в наиболее компактной форме. Кодер канала отвечает за защиту сообщения от помех в канале связи. На схеме нет модулятора, который переводит цифровой сигнал в форму, соответствующую физической среде передачи. В данном курсе будет рассматриваться кодирование дискретных источников (source encoder).

Под каналом может пониматься не только передача (кабель), но и хранение (жесткий диск).

2.2 Практическое применение

Практическое применение: архивирование данных (ZIP, RAR, 7-Zip), сжатие речи, сжатие звука, сжатие изображений, сжатие видео.

Архивация данных подразумевает сжатие без потерь. Остальные виды сжатия допускают потери (уменьшается качество).

Пропускная способность сетей растет медленней, чем объем данных, который нужно передавать. Поэтому актуальность сжатия данных сохраняется.

2.3 Дискретные ансамбли

Дискретное множество содержащее конечное число элементарных событий:

$$X = \{x\}.$$

Множество чисел, которое задает **распределение вероятностей**:

$$\{p(x)\}, p(x) > 0, \sum p(x) = 1,$$

где $p(x)$ — вероятность исхода x .

Дискретный ансамбль:

$$X = \{x, p(x)\}.$$

Множество всевозможных подмножеств X :

$$\Omega = \{A\}.$$

Вероятность сложного события A :

$$P(A) = \sum_{x \in A} p(x), \quad A \in \Omega.$$

Элементарные события несовместны, поэтому вероятности просто складываются.

Вероятность совместного наступления зависимых событий:

$$P(AB) = P(B) \cdot P(A | B).$$

В общем случае:

$$P(A_1 \cdots A_n) = P(A_1)P(A_2 | A_1)P(A_3 | A_1A_2) \cdots .$$

Сама зависимость одного события от другого выражается **формулой условной вероятности**: $P(A | B) = \frac{P(AB)}{P(B)}$. Если $P(B) = 0$, то $P(A | B) = 0$.

Пример зависимого события: достать черный шар из корзины, если из нее уже достали белый шар.

События $A, B \in \Omega$ независимы, если их пересечение:

$$P(AB) = P(A)P(B).$$

Вероятности таких событий:

$$P(A | B) = P(A), \quad P(B | A) = P(B).$$

Вероятность объединения событий (произойдет хотя бы одно событие, учитывается совместный исход):

$$P(A \cup B) = P(A) + P(B) - P(AB),$$

$$P\left(\bigcup_{m=1}^M A_m\right) \leq \sum_{m=1}^M P(A_m).$$

Рассмотрим графы с вероятностями переходов:

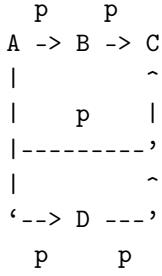
$$\begin{array}{ccc} p & & p \\ A & \rightarrow & B & \rightarrow & C \end{array}$$

Нужно 2 перехода, поэтому вероятность перейти из A в C: p^2 .

$$\begin{array}{ccc} p & & p \\ A & \rightarrow & B & \rightarrow & C \\ | & & & \sim & \\ '-----', & & & & \\ p & & & & \end{array}$$

Пути 2, но переход в C совершился только по одному: $P(AC) = p^2 + p - p^3$. Для

проверки можно взять $p = 1$.



Такой случай сложный, но его можно легко посчитать по формуле включений-исключений:

$$\begin{aligned}
 P(AC) &= P(ABC) \cup P(AC) \cup P(ADC) = \\
 &= P(ABC) + P(AC) + P(ADC) - \\
 &\quad - P(ABC) \cap P(AC) - P(ABC) \cap P(ADC) - P(AC) \cap P(ADC) + \\
 &\quad + P(ABC) \cap P(AC) \cap P(ADC) = 2p^2 + p - 2p^3 - p^4 + p^5.
 \end{aligned}$$

Пусть даны M несовместных событий H_1, \dots, H_M , таких что $P(\bigcup_{m=1}^M H_m) = 1$. Тогда вероятность произвольного события A считается по формуле **полней вероятности**:

$$P(A) = \sum_{m=1}^M P(A \mid H_m) P(H_m).$$

Формула **апостериорной вероятности (Байеса)**:

$$P(H_j \mid A) = \frac{P(AH_j)}{P(A)} = \frac{P(A \mid H_j)P(H_j)}{\sum_{m=1}^M P(A \mid H_m)P(H_m)}.$$

Вероятность того, что предпосылка произошла, если произошло следствие.

Пусть у первого грузчика 200 коробок, у второго — 300, у третьего — 700. Вероятность, что первый разбьет коробку 0,1, второй — 0,2, третий — 0,9. Найти вероятность, что коробка разбита вторым грузчиком. Пусть A — коробка разбита, H — коробку нес грузчик:

$$P(H_2 \mid A) = \frac{0,2 \cdot \frac{300}{1200}}{0,1 \cdot \frac{200}{1200} + 0,2 \cdot \frac{300}{1200} + 0,9 \cdot \frac{700}{1200}} = \frac{0,05}{0,592} \approx 0,084.$$

Произведение ансамблей $X = \{x, p_X(x)\}$ и $Y = \{y, p_Y(y)\}$:

$$XY = \{(x, y), p_{XY}(x, y)\}.$$

Условное распределение вероятностей:

$$p(x \mid y) = \begin{cases} \frac{p(x,y)}{p(y)} & \text{if } p(y) \neq 0, \\ 0 & \text{иначе.} \end{cases}$$

Ансамбли X и Y независимы, если

$$p(x, y) = p(x)p(y), \quad x \in X, \quad y \in Y.$$

2.4 Измерение информации

Интуитивный подход: количество информации равняется затратам, необходимым для передачи или хранения данных (буква текста равняется 8 битам).

Информацию логично измерять через вероятность. Если одно событие происходит регулярно, то оно перестает быть информативным. Редкие события несут много информации.

Пусть $X = \{x, p(x)\}$ — ансамбль, $\mu(x)$ — мера (количество) информации в x .

- $\mu(x) \geq 0$.
- $\mu(x)$ — функция от $p(x)$.
- Монотонность: если $x, y \in X$, $p(x) \geq p(y)$, тогда $\mu(x) \leq \mu(y)$.
- Аддитивность: если x и y независимы, тогда $\mu(x, y) = \mu(x) + \mu(y)$.
- $\mu(p(x)^k) = k\mu(p(x))$: если проходит серия одного события, то информация каждого складывается.

Из этого следует формула **собственной информации** в событии x :

$$I(x) = -\log p(x), \quad x \in X.$$

Основание логарифма может быть любым. Если взять 2, то информация будет измеряться в битах.

Энтропия — средняя информативность ансамбля:

$$H(X) = E[I(x)] = \sum_x p(x)I(x) = -\sum_x p(x)\log p(x).$$

Свойства энтропии:

1. $H(X) \geq 0$.
2. $H(X) \leq \log |X|$. Логарифм от количества сообщений. Равенство, если все элементы X равновероятны.

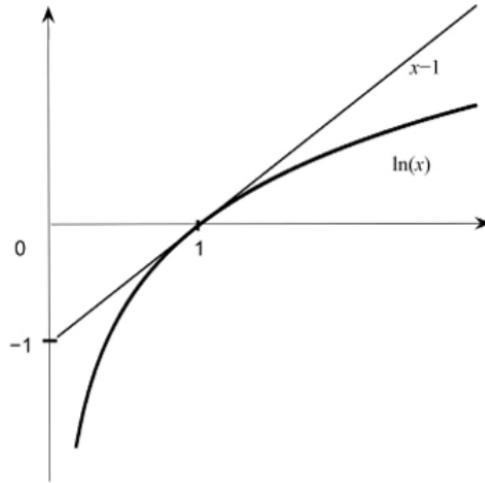
Доказательство

$$\begin{aligned} H(X) - \log |X| &= -\sum_{x \in X} p(x)\log p(x) - \sum_{x \in X} p(x)\log |X| = \\ &= \sum_{x \in X} p(x)\log \frac{1}{p(x)|X|} \leq \\ &\leq \log e \left[\sum_{x \in X} p(x) \left(\frac{1}{p(x)|X|} - 1 \right) \right] = \\ &= \log e \left(\sum_{x \in X} \frac{1}{|X|} - \sum_{x \in X} p(x) \right) = 0 \end{aligned}$$

На первом шаге пользуемся тем, что сумма вероятностей равна 1. На втором шаге собираем логарифм и переворачиваем его число, чтобы избавиться от

минуса. На третьем шаге используем свойство логарифма:

$$\begin{aligned}\ln x &\leq x - 1 \\ e^{\ln x} &= e^{x-1} \\ x &= e^{x-1} \\ \log x &= \log e^{x-1} = (x-1) \log e\end{aligned}$$



чтобы преобразовать логарифм, затем вынесем общий множитель $\log e$ за сумму. На четвертом шаге раскроем круглые скобки и заменим одну сумму двумя. Первая сумма равна 0, так как суммируем $|X|$ раз.

3. $X = \{x, p(x)\}$ и $Y = \{y = f(x), p(y)\}$, тогда $H(Y) \leq H(X)$, с равенством, если f обратима.
4. Если X и Y независимы, тогда $H(XY) = H(X) + H(Y)$.
5. $H(X)$ — выпуклая вверх функция распределения вероятностей на элементах ансамбля X .
6. Пусть $X = \{x, p(x)\}$ и $A \subseteq X$. Введем ансамбль $X' = \{x, p'(x)\}$ и $p'(x)$ как:

$$p'(x) = \begin{cases} \frac{P(A)}{|A|}, & x \in A, \\ p(x), & x \notin A \end{cases}$$

тогда $H(X') \geq H(X)$. В первом кейсе усреднение части ансамбля.

7. Если для двух ансамблей X и Y распределения вероятностей отличаются только порядком следования элементов, то $H(X) = H(Y)$.

3 Выпуклые функции. Условная энтропия. Стационарные источники

3.1 Выпуклые функции

Множество вещественных векторов $\mathbb{R} = \{x\}$ выпукло, если $\forall x, x' \in \mathbb{R}$ и $\forall \alpha \in [0, 1]$ вектор $y = \alpha x + (1 - \alpha)x'$ принадлежит \mathbb{R} . Можно представить вектор в виде

точки. Тогда множество называется выпуклым, если любые две точки множества задают отрезок, все точки которого принадлежат этому множеству.

Теорема Множество вероятностных векторов длины M выпукло.

Доказательство Рассмотрим два распределения вероятностей $p = \{p_1, \dots, p_M\}$ и $p' = \{p'_1, \dots, p'_M\}$. Рассмотрим векторы вида

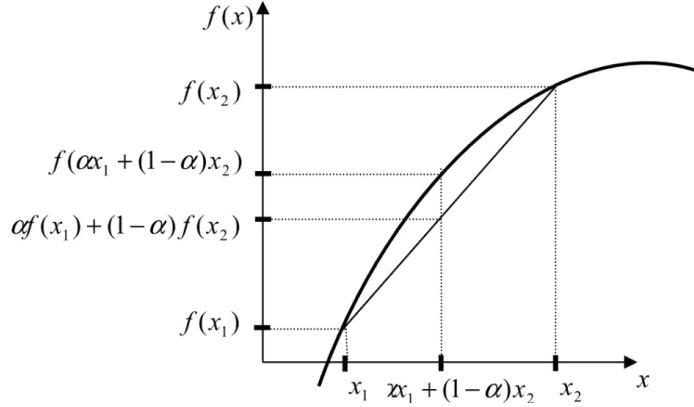
$$q = \alpha p + (1 - \alpha)p',$$

где $q_i \geq 0$, а сумма компонент:

$$\sum_{i=1}^M q_i = \alpha \sum_{i=1}^M p_i + (1 - \alpha) \sum_{i=1}^M p'_i = \alpha + 1 - \alpha = 1.$$

Функция $f(x)$ выпуклая, если $\forall x, x' \in R$, где R — выпуклая область, и $\forall \alpha \in [0, 1]$ выполняется неравенство: $f(\alpha x + (1 - \alpha)x') \geq \alpha f(x) + (1 - \alpha)f(x')$.

Доказательство



Значение функции на прямой выводится через уравнение прямой по двум точкам.

Для случая функции одной переменной:

$$f(\alpha x_1 + (1 - \alpha)x_2) \geq \alpha f(x_1) + (1 - \alpha)f(x_2).$$

Если функция выпукла вниз, то неравенство меняется в обратную сторону.

Теорема Пусть $f(x)$ — выпуклая \cap функция вектора x , определенная на выпуклой области R , и пусть константы $a_1, \dots, a_M \in [0, 1]$ такие, что $\sum_{m=1}^M a_m = 1$. Тогда $\forall x_1, \dots, x_M \in R$ справедливо неравенство:

$$f\left(\sum_{m=1}^M a_m x_m\right) \geq \sum_{m=1}^M a_m f(x_m).$$

Если α_m означает вероятность x_m , то получим **неравенство Йенсена** (функция от мат. ожидания больше либо равна мат. ожидания от функции):

$$f(E\{x\}) \geq E\{f(x)\}.$$

Свойства выпуклых функций:

- сумма выпуклых функций выпукла,
- произведение вып. функции и положительной константы является вып. функцией,
- линейная комбинация (сумма с коэффициентами) с неотрицательными коэффициентами — выпуклая функция (обобщение первого свойства).

Доказательство 5 свойства энтропии.

$$H(p) = - \sum_{m=1}^M p_m \log p_m$$

Рассмотрим слагаемые $f_m(p) = -p_m \log p_m$. Если $f_m(p)$ выпуклая, то их сумма будет выпуклой функцией.

Вторая производная $f_m''(p) = \frac{-(\log e)}{p_m}$. $f_m''(p) < 0 \forall p_m \in (0, 1)$.

Пример $X = \{0, 1\}$, $p(1) = p$, $p(0) = 1 - p = q$. Энтропия двоичного ансамбля:

$$H(X) = -p \log p - q \log q \triangleq \eta(p)$$

Последнее равенство означает равенство по определению. Просто ввели новое обозначение для двоичной энтропии.

$$\begin{aligned} \eta'(p) &= -\log p + \log(1-p) \\ \eta''(p) &= -\log \frac{e}{p} - \log \frac{e}{1-p} < 0 \end{aligned}$$

Следовательно, $H(X)$ выпуклая. Максимум энтропии равен 1 (при основании логарифма 2).

Рассмотрим случай, когда $p = 0$:

$$H(X) = -0 \times -\infty - 1 \times 0.$$

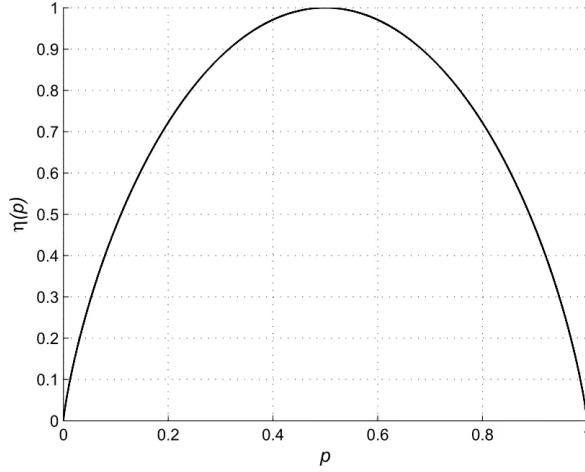
Неопределенность первого слагаемого решим с помощью правила Лопиталя:

$$\lim_{p \rightarrow 0} -\frac{\log p}{p^{-1}} = \lim_{p \rightarrow 0} \frac{p^2}{p \ln a} = \lim_{p \rightarrow 0} \frac{p}{\ln a} = 0$$

Получаем, что $H(X) = 0$.

Если $p = 1$, то аналогичными рассуждениями доказывается, что $H(X) = 0$.

Не сложно получить график энтропии двоичного ансамбля:



В Теории информации битами измеряется среднее количество информации в источнике. Может получиться, что в среднем в одном символе ансамбля или одном сообщении источника меньше одного бита информации. В классическом, инженерном смысле бит является единицей информации, но в данном случае это не так.

Видно, что чем выше неопределенность событий (чем меньше разность между их вероятностями), тем больше их энтропия, т.е средняя информативность.

Доказательство свойства 6 энтропии.

Обозначим $\tilde{p} = (\frac{p_1+p_2}{2}, \frac{p_1+p_2}{2}, p_3, \dots, p_M)$

Докажем, что $H(\tilde{p}) \geq H(p)$

$$\begin{aligned} p' &= p = (p_1, p_2, p_3, \dots, p_M) \\ p'' &= (p_2, p_1, p_3, \dots, p_M) \end{aligned}$$

Заметим, что: $H(p') = H(p'') = H(p)$ $\tilde{p} = \frac{p' + p''}{2}$

Из выпуклости энтропии следует, что

$$H(\tilde{p}) = H\left(\frac{p' + p''}{2}\right) \geq \frac{1}{2}H(p') + \frac{1}{2}H(p'') = H(p)$$

Равенство будет, если события равновероятны.

Условная собственная информация сообщения x при фиксированном y :

$$I(x | y) = -\log p(x | y)$$

Условная энтропия X при заданном $y \in Y$:

$$H(X | y) = - \sum_{x \in X} p(x | y) \log p(x | y)$$

Условная энтропия X при фиксированном ансамбле Y :

$$H(X | Y) = - \sum_{y \in Y} \left(p(y) \sum_{x \in X} p(x | y) \log p(x | y) \right) = - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(x | y)$$

Свойства условной энтропии:

1. $H(X | Y) \geq 0$
2. $H(X | Y) \leq H(X)$, равенство, если X и Y независимы

Доказательство:

$$\begin{aligned} H(X | Y) - H(X) &= - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(x | y) + \sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(x) = \\ &= \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{p(x)}{p(x | y)} \leq \\ &\leq \sum_{x \in X} \sum_{y \in Y} p(x, y) \left(\frac{p(x)}{p(x | y)} - 1 \right) \log e = \\ &= \left(\sum_{x \in X} \sum_{y \in Y} p(y)p(x) - \sum_{x \in X} \sum_{y \in Y} p(x, y) \right) \log e = 0 \end{aligned}$$

На первом шаге $H(X)$ заменили ее формулой, затем расписали $p(x)$ по формуле полной вероятности как $\sum_{y \in Y} p(y)p(x | y) = \sum_{y \in Y} p(x, y)$.

На третьем шаге нужно вспомнить доказательство 2 свойства энтропии.

3. $H(XY) = H(X) + H(Y | X) = H(Y) + H(X | Y)$
4. $H(X | YZ) \leq H(X | Y)$, равенство, если X и Z условно независимы для всех $y \in Y$. Чем больше условий, тем меньше информации.
5. $H(X_1 \cdots X_n) = H(X_1) + H(X_2 | X_1) + H(X_3 | X_1X_2) + \cdots + H(X_n | X_1, \dots, X_{n-1})$. В качестве $X_1 \cdots X_n$ может быть текст длины n , где каждую букву определяет ансамбль.
6. $H(X_1 \cdots X_n) \leq \sum_{i=1}^n H(X_i)$ равенство, если X_1, \dots, X_n являются совместно независимыми.

Пусть загадано слово. Если угадывать первую букву в этом слове, то она может быть почти любой. Если нам известно начало слова, например, МОСКВ, то становится очевидней, какая буква следующая. Поэтому энтропия первых букв больше, чем последних. Энтропию первой буквы можно обозначить $H(X)$, второй — $H(Y | X)$ и т.д.

Доказательство 3 свойства энтропии:

Используя свойство 3 условной энтропии:

$$H(XY) = \underbrace{H(X | Y)}_{\geq 0} + H(Y) = \underbrace{H(Y | X)}_{= 0} + H(X)$$

Поскольку $f(x)$ определена для каждого x , получим, что $H(Y | X) = 0$, $H(X | Y) \geq 0$

Так как знаем функцию, неопределенность Y исчезает. Но функция может возвращать одно значение для разных x , поэтому для X имеется неопределенность.

3.2 Дискретные источники

Дискретный источник — это устройство, которое в каждый момент времени выбирает одно сообщение из дискретного множества. Например, букву из алфавита.

Если множество значений времени также дискретно, то источник называется **дискретным по времени**.

Источник считается заданным, если известна его вероятностная модель. Должна быть определена вероятностная модель случайного процесса генерирования случайных сообщений на выходе источника.

Случайный процесс (также называемый стохастическим) — набор случайных величин, проиндексированных множеством, которое часто обозначает разные моменты времени. Множество индексов может быть или множеством натуральных чисел (случайный процесс с дискретным временем), или множеством вещественных чисел (случайный процесс с непрерывным временем). Случайные величины могут иметь непрерывное или дискретное **пространство состояний** (пространство возможных результатов в каждый момент времени).

Дискретный источник задан, если для $n = 1, 2, \dots$ (длина угадываемого слова) и $i = 0, 1, 2, \dots$ (номер угадываемого слова) известна вероятность $p(x_{i+1}, x_{i+2}, \dots, x_{i+n})$ случайной последовательности из $\{X_{i+1}X_{i+2}\dots X_{i+n}\}$, которая начинается с индекса $i + 1$ и имеет длину n , где $x_j \in X_j$, $j = i + 1, \dots, i + n$. Обычно рассматривается случай, когда $X_j = X$ для всех j , то есть алфавит для каждого символа в одном сообщении фиксированный.

Пусть $x = (x_1, x_2, \dots, x_n, \dots)$ — дискретный случайный процесс (генерируемая последовательность букв). Рассмотрим случайный вектор $x_{j+1}^{j+n} = (x_{j+1}, x_{j+2}, \dots, x_{j+n})$ (окно на генерируемой последовательности с длиной n и смещением j от начала, слово).

Стационарность процесса означает, что для любого n и j , $p(x_{j+1}^{j+n})$ не зависит от сдвига во времени j , т.е., $p(x_{j+1}^{j+n}) = p(x_1^n)$.

Источник называют **дискретным без памяти**, если (следующая буква не зависит от предыдущей)

$$p(x_1, x_2, \dots, x_n) = \prod_1^n p(x_i)$$

3.2.1 Цепь Маркова

Дискретный случайный процесс называется **цепью Маркова** порядка s , если для любого n и $x = (x_1, \dots, x_n) \in X^n$ выполняется следующее равенство:

$$p(x) = p(x_1, \dots, x_s)p(x_{s+1} | x_1, \dots, x_s)p(x_{s+2} | x_2, \dots, x_{s+1}) \cdots p(x_n | x_{n-s}, \dots, x_{n-1})$$

Другими словами, первые s букв независимы, затем каждая следующая буква зависит от s предыдущих.

Для цепи Маркова справедливо равенство:

$$p(x_n | x_1, \dots, x_{n-1}) = p(x_n | x_{n-s}, \dots, x_{n-1})$$

x_t — состояние марковской цепи в момент времени t . Марковская цепь порядка s задается начальным распределением вероятностей первых s значений (состояний) и условными вероятностями вида $p(x_n | x_{n-s}, \dots, x_{n-1})$ для всевозможных последовательностей x_{n-s}, \dots, x_{n-1} . Если эти условные вероятности не изменяются при сдвиге во времени, то марковская цепь называется **однородной**.

Чтобы определить условную вероятность буквы, нужно знать s предыдущих букв. Таким образом порядок марковской цепи определяет память источника.

Марковская цепь порядка $s = 1$ с состояниями $X = \{0, 1, \dots, M-1\}$ определяется начальным распределением $\{p(x_1), x_1 \in X\}$ и условными вероятностями

$$\pi_{ij} = P(x_t = j | x_{t-1} = i), \quad i, j = 0, 1, \dots, M-1$$

Матрица переходных вероятностей (вероятность, что текущее состояние j , а предыдущее было i):

$$\Pi = \begin{pmatrix} \pi_{00} & \pi_{01} & \cdots & \pi_{0,M-1} \\ \pi_{10} & \pi_{11} & \cdots & \pi_{1,M-1} \\ \cdots & \cdots & \cdots & \cdots \\ \pi_{M-1,0} & \pi_{M-1,1} & \cdots & \pi_{M-1,M-1} \end{pmatrix}$$

Обозначим через $p_t = (p_t(0), \dots, p_t(M-1))$ стохастический вектор, компоненты которого — вероятности состояний цепи Маркова в момент времени t , где $p_t(i)$, $i = 0, 1, \dots, M-1$ — вероятность состояния i в момент времени t .

Из формулы полной вероятности следует:

$$p_{t+1}(i) = \sum_{j=0}^{M-1} p_t(j) \pi_{ji}$$

В матричном виде:

$$\mathbf{p}_{t+1} = \mathbf{p}_t \Pi$$

Для произвольного числа шагов n :

$$\mathbf{p}_{t+n} = \mathbf{p}_t \Pi^n$$

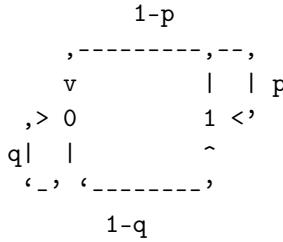
Из формулы следует, что распределение вероятностей в момент времени t зависит от величины t и от начального распределения p_1 . Отсюда следует, что в общем случае рассматриваемый случайный процесс нестационарен. Однако, если существует стохастический вектор p , так что

$$\mathbf{p} = \mathbf{p} \Pi$$

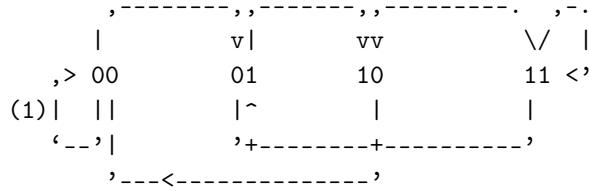
то выбрав $\mathbf{p}_1 = \mathbf{p}$ мы получим стационарный процесс. Вектор \mathbf{p} , удовлетворяю-

щий называется **стационарным распределением вероятностей** для марковской цепи с матрицей переходных вероятностей Π .

Рассмотрим марковскую цепь с 2 состояниями в качестве источника с памятью 1:



Увеличим память до 2:



В кружках написаны последние 2 символа, так как каждый следующий зависит от предыдущих 2

Энтропия символа x_t (из ансамбля $X_t = X$) сгенерированного в момент времени t не зависит от t ($H(X_t) = H(X)$) и называется **одномерной энтропией источника** (процесса). Обозначим ее как $H_1(X)$.

$H_1(X)$ не учитывает зависимость между символами, порожденными источником. Это не относится к марковскому источнику.

3.3 Энтропия на сообщение

Рассмотрим $x = (x_1, x_2, \dots, x_n)$ из $X_1 X_2 \cdots X_n = X^n$. Энтропия $H(X_1 X_2 \cdots X_n) = H(X^n)$ называется **n-мерной энтропией** процесса.

Энтропия на символ для последовательности длины n определяется как:

$$H_n(X) = \frac{H(X^n)}{n},$$

То есть берем всю последовательность, сгенерированную источником, и делим ее на блоки длины n . Делим энтропию блока на количество символов в нем.

другой способ:

$$H(X_n | X_1, \dots, X_{n-1}) = H(X | X^{n-1}).$$

Энтропия на сообщение:

$$\lim_{n \rightarrow \infty} H_n(X)$$

$$\lim_{n \rightarrow \infty} H(X | X^n)$$

Проблема блочного кодирования в том, что первые символы в блоке будут плохо кодироваться, так как мало информации о предыдущих символах. Разумно не использовать маленький размер блока.

Для дискретного стационарного процесса (источника)

1. $H(X | X^n)$ не возрастает с увеличением n .

Следует из невозрастания энтропии с увеличением числа условий.

2. $H_n(X)$ не возрастает с увеличением n .

$$\begin{aligned} H(X^{n+1}) &= H(X_1 \cdots X_n X_{n+1}) = \\ &= H(X_1 \cdots X_n) + H(X_{n+1} | X_1, \dots, X_n) \leq \\ &\leq nH_n(X) + H_n(X) = \\ &= (n+1)H_n(X). \end{aligned}$$

На третьем шаге для второго слагаемого из предыдущего шага используется свойство 3.

$$H(X^{n+1}) \leq (n+1)H_n(X) \implies \frac{H(X^{n+1})}{n+1} = H_{n+1}(X) \leq H_n(X)$$

3. $H_n(X) \geq H(X | X^{n-1})$

$$H(X^n) = H(X) + H(X | X^1) + \cdots + H(X | X^{n-1}) \geq nH(X | X^{n-1}) \geq nH(X | X^n)$$

4. $\lim_{n \rightarrow \infty} H_n(X) = \lim_{n \rightarrow \infty} H(X | X^n)$

Доказательство

Из 3 свойства следует, что $\lim_{n \rightarrow \infty} H_n(X) \geq \lim_{n \rightarrow \infty} H(X | X^n)$

Для $m < n$:

$$\begin{aligned} H(X^n) &= H(X_1 \cdots X_n) = \\ &= H(X_1 \cdots X_m) + H(X_{m+1} | X_1, \dots, X_m) + \cdots + H(X_n | X_1, \dots, X_{n-1}) \leq \\ &\leq mH_m(X) + (n-m)H(X | X^m) \end{aligned}$$

После деления обеих частей на n :

$$\lim_{n \rightarrow \infty} H_n(X) \leq H(X | X^m) \text{ для любого } m.$$

Справа от неравенства:

$$\lim_{n \rightarrow \infty} \frac{mH_m(X)}{n} + \frac{nH(X | X^m)}{n} - \frac{H(X | X^m)}{n} \xrightarrow[0]{\quad}$$

Устремляем $m \rightarrow \infty$:

$$\lim_{n \rightarrow \infty} H_n(X) \leq \lim_{m \rightarrow \infty} H(X | X^m)$$

4 Энтропия на сообщение дискретного источника. Префиксные коды

4.1 Энтропия на сообщение

Обозначим $H_\infty(X) = \lim_{n \rightarrow \infty} H_n(X)$, $H(X | X^\infty) = \lim_{n \rightarrow \infty} H(X | X^n)$, тогда

$$H_\infty(X) = H(X | X^\infty)$$

Два способа кодирования:

- Расширение алфавита: буквы становятся последовательностями исходных букв длины n .
- Учет зависимости текущей буквы от n предшествующих букв.

Энтропии для дискретного стационарного источника без памяти:

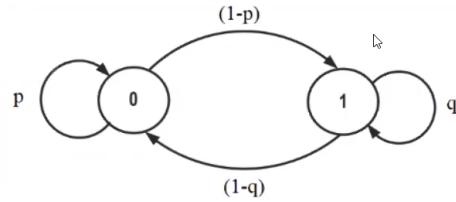
- $H(X_1 \cdots X_n) = H(X_1) + \cdots + H(X_n)$ (нет памяти)
- $H(X^n) = nH(X)$ (один алфавит)
- $H_n(X) = H(X)$ (из определения)
- $H_\infty(X) = H(X)$
- $H(X | X^n) = H(X_{n+1} | X_1, \dots, X_n) = H(X)$,
- $H(X | X^\infty) = H(X)$

Отсюда не следует, что для такого источника нужно кодировать каждую букву независимо от других.

Энтропии для марковского источника:

- $H(X | X^n) = H(X_{n+1} | X_1, \dots, X_n) = H(X_{n+1} | X_{n-s+1}, \dots, X_n) = H(X | X^s)$
- $H(X | X^\infty) = H(X | X^s)$
- $H(X^n) = H(X_1 \cdots X_s X_{s+1} \cdots X_n)$
 $= H(X_1 \cdots X_s) + H(X_{s+1} \cdots X_n | X_1, \dots, X_s)$
- $H(X_{s+1} \cdots X_n | X_1, \dots, X_s) = H(X_{s+1} | X_1, \dots, X_s) + H(X_{s+2} | X_2, \dots, X_{s+1}) + \cdots + H(X_n | X_{n-s}, \dots, X_{n-1}) = (n-s)H(X | X^s)$
- $\frac{H(X^n)}{n} = \frac{sH_s(X)}{n} + \frac{(n-s)H(X | X^s)}{n}$

Рассмотрим марковский источник (состояния и вероятности переходов):



Выведем энтропию на символ:

$$H(X) = -\pi_0 \log \pi_0 - \pi_1 \log \pi_1$$

Здесь обозначили вероятность, что находимся в состоянии 0, как π_0 . Найдем π_0 и π_1 :

$$\begin{aligned} \pi_0 &= \pi_0 p + \pi_1 (1 - q) \\ \pi_1 &= 1 - \pi_0 \\ \pi_0 &= \pi_0 p + (1 - \pi_0) - q(1 - \pi_0) \\ \pi_0 - \pi_0 p + \pi_0 - \pi_0 q &= 1 - q \\ \pi_0 &= \frac{1 - q}{2 - p - q} \\ \pi_1 &= \frac{1 - p}{2 - p - q} \end{aligned}$$

Если цепь будет большой, то аналитический способ нахождения вероятностей может оказаться слишком сложным. Найдем стационарное распределение путем моделирования. Произвольно зададим начальное распределение p_1 , например, пусть мы находимся в состоянии 1: $p_1 = (0, 1)$. Матрица переходных вероятностей для рассматриваемой цепи:

$$\Pi = \begin{pmatrix} p & 1 - p \\ 1 - q & q \end{pmatrix}$$

Программа:

```

p = 0.25
q = 0.35
P = [[p, 1 - p], [1 - q, q]]

p1 = [0, 1]

pn = [p1]
for _ in range(10):
    p1 = [p1[0] * P[0][0] + p1[1] * P[1][0], \
          p1[0] * P[0][1] + p1[1] * P[1][1]]
    pn.append(p1)

import matplotlib.pyplot as plot

```

```

plot.plot(range(11), [p[0] for p in pn])
plot.plot(range(11), [p[1] for p in pn])
plot.show()

```

Конечное распределение будет приближено к стационарному (или тому, которое нашли аналитическим путем).

Надем условную энтропию. Для этого воспользуемся формулой, приведенной ранее:

$$\begin{aligned}
H(X|X) &= -[\pi_0(p(0|0)\log p + p(1|0)\log p) + \pi_1(p(0|1)\log p + p(1|1)\log p)] = \\
&= -[\pi_0(p\log p + (1-p)\log(1-p)) + \pi_1((1-q)\log(1-q) + q\log q)] = \\
&= \pi_0\eta(p) + \pi_1\eta(q)
\end{aligned}$$

Условные вероятности определяются вероятностями переходов. Вероятность нахождения в определенном состоянии определяется из стационарного распределения.

Найдем энтропию на символ из блока. Наш источник имеет память 1, поэтому условная энтропия не могут быть глубже 1. Если бы у источника не было памяти, то в качестве энтропии на символ, мы бы просто взяли $H(X)$.

$$H_2(X) = \frac{H(X) + H(X|X)}{2}, H_3(X) = \frac{H(X) + 2H(X|X)}{3}$$

Просто $H(X)$ игнорирует модель источника. Это самый плохой случай. $H(X|X)$ — меньшая и лучшая величина, лучшая так как она учитывает модель. Чем больше у нас размер блока, тем меньше вклад худшей величины и тем лучше значение энтропии на символ.

4.2 Неравномерное побуквенное кодирование

Рассмотрим дискретный источник без памяти: $X = \{1, \dots, M\}$ (источник генерирует числа от 1 до M), $\{p_1, \dots, p_M\}$ (соответствующие вероятности). $C = \{c_1, \dots, c_M\}$ — кодовые слова длины l_1, \dots, l_M .

Средняя длина кодового слова:

$$\bar{l} = E[l_i] = \sum_{i=1}^M p_i l_i$$

$H(X)$ — нижняя граница для \bar{l} (доказательство ниже).

Длины кодовых слов могут отличаться, поэтому кодирование неравномерное (variable length coding).

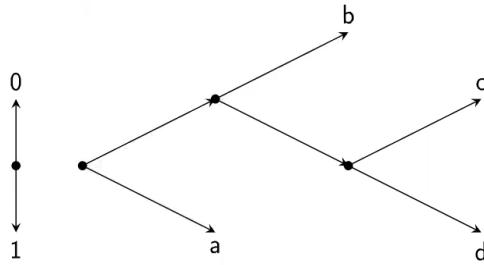
Рассмотрим код Морзе:

Для однозначного декодирования кода необходимы разделители (паузы) между

Буква	Кодовое слово
e	.
a	. -
j	. - - -
q	- - . -

кодовыми словами.

Рассмотрим двоичное кодовое дерево:



Буква	Кодовое слово
a	1
b	00
c	010
d	011

Такой код называется **префиксным**. Здесь нет проблемы, которая была в коде Морзе. Вводить дополнительный символ-разделитель не требуется.

Свойства префиксного кода:

- ни одно кодовое слово не является началом другого кодового слова,
- префиксный код является однозначно декодируемым,
- если только листья двоичного дерева соответствуют кодовым словам, то код является префиксным,
- однозначно декодируемый код не обязательно является префиксным,
- древовидный код является префиксным.

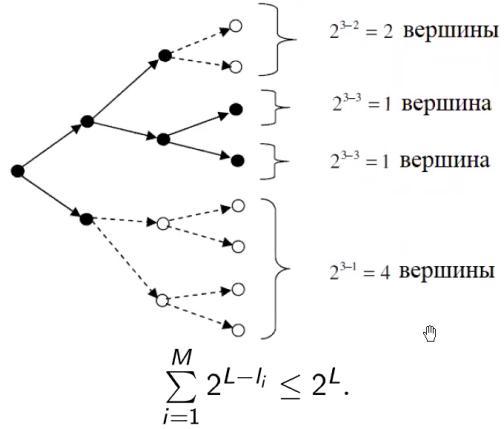
Необходимое и достаточное условие существования префиксного кода (неравенство Крафта):

$$\sum_{i=1}^M 2^{-l_i} \leq 1$$

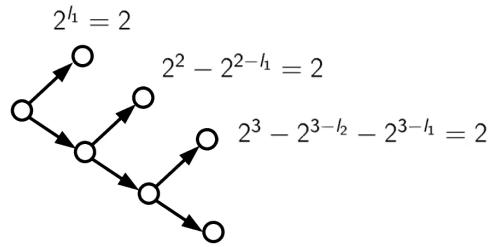
Доказательство необходимости Пусть $L \geq \max l_i$, тогда если достроить двоичное дерево до полного, справедливо неравенство:

$$\sum_{i=1}^M 2^{L-l_i} \leq 2^L.$$

Меньше, если одну букву удалили. Если поделить на правую часть, то получим неравенство Крафта.



Доказательство достаточности Длину кодового слова определяет уровень буквы в двоичном дереве. Если в дереве нет листьев (ни одна буква не кодируется), то для кода длины l есть 2^l вариантов (количество узлов на уровне l). Количество вариантов будет уменьшаться в зависимости от количества листов на верхних уровнях, но для любой длины всегда найдется хотя бы один свободный узел.



Для любого однозначно декодируемого двоичного кода объемом M с длинами кодовых слов l_1, \dots, l_M справедливо неравенство:

$$\sum_{i=1}^M 2^{-l_i} \leq 1.$$

Прямая теорема неравномерного побуквенного кодирования Для ансамбля $X = \{x, p(x)\}$ с энтропией $H(X) = H$ существует побуквенный неравномерный префиксный код со средней длиной кодовых слов $\bar{l} < H + 1$.

Доказательство Пусть $l_i = \lceil -\log p_i \rceil$ (логарифм округлим вверх). Тогда $\sum_{i=1}^M 2^{-l_i} = \sum_{i=1}^M 2^{-\lceil -\log p_i \rceil} \leq \sum_{i=1}^M 2^{\log p_i} = 1 \implies$ такой префиксный код существует.

$$\bar{l} = \sum_{m=1}^M p_m l_m < \sum_{m=1}^M p_m (-\log p_m + 1) < H + 1. \quad \square$$

Обратная теорема неравномерного побуквенного кодирования Для любого однозначно декодируемого кода для дискретного источника $\{X, p(x)\}$ с энтропией H справедливо $\bar{l} \geq H$.

Доказательство

$$\begin{aligned}
H - \bar{l} &= - \sum_{x \in X} p(x) \log p(x) - \sum_{x \in X} p(x)l(x) = \\
&= \sum_{x \in X} p(x) \log \frac{2^{-l(x)}}{p(x)} \leq \log e \sum_{x \in X} p(x) \left(\frac{2^{-l(x)}}{p(x)} - 1 \right) \leq \\
&\leq \log e \left(1 - \sum_{x \in X} p(x) \right) = 0
\end{aligned}$$

На втором шаге преобразовали: $-l(x) = \log 2^{-l(x)}$.

Свойства оптимального побуквенного кода:

- если $p_i < p_j$, то $l_i \geq l_j$ (чем чаще встречается кодовое слово, тем меньше должна быть его длина),
- есть кодовые слова, которые имеют одинаковую длину $l_M = \max_m l_m$,
- среди кодовых слов длиной $l_M = \max_m l_m$ найдутся два слова, различающиеся только в одном последнем символе.
- Пусть $p_1 \leq p_2 \leq \dots \leq p_M$. Для ансамбля $X = \{1, \dots, M\}$ и кода С, удовлетворяющего свойствам 1-3, введем ансамбль $X' = \{1, \dots, M-1\}$, сообщениями которого приписаны вероятности $\{p'_1, \dots, p'_{M-1}\}$ так, что

$$\begin{aligned}
p'_1 &= p_1, \\
p'_2 &= p_2, \\
p'_{M-1} &= p_{M-1} + p_M.
\end{aligned}$$

Из кода С построим код C' для ансамбля X' , приписав сообщениям x'_1, \dots, x'_{M-2} те же кодовые слова, что и в коде, т.е. $c'_i = c_i$, а сообщению x'_{M-1} слово c'_{M-1} , как общую часть слов C_{M-1} и C_M .

Тогда если C' оптимален для X' , то код C оптимален для X .

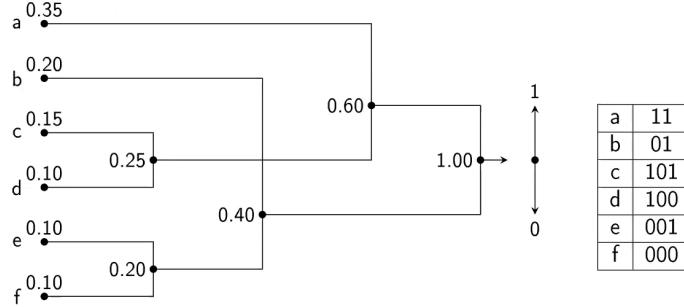
Доказательство

$$l_m = \begin{cases} l'_m & m \leq M-2, \\ l'_{M-1} + 1 & m = M-1, M. \end{cases}$$

$$\begin{aligned}
\bar{l} &= \sum_{m=1}^M p_m l_m = \sum_{m=1}^{M-2} p_m l_m + p_{M-1} l_{M-1} + p_M l_M = \\
&= \sum_{m=1}^{M-2} p_m l_m + (p_{M-1} + p_M)(l'_{M-1} + 1) = \\
&= \sum_{m=1}^{M-2} p'_m l'_m + p'_{M-1} l'_{M-1} + p_{M-1} + p_M = \\
&= \sum_{m=1}^{M-1} p'_m l'_m + p_{M-1} + p_M = \bar{l}' + p_{M-1} + p_M
\end{aligned}$$

Во второй строке смотрим на кейсы l_m , в третьей строке раскрываем скобки.

Код Хаффмана позволяет построить оптимальный побуквенный код:



Имеются исходные символы, которым сопоставлены вероятности, с которыми их генерирует источник.

Находим 2 сообщения с минимальными вероятностями и соединяем их в новую вершину, вероятность которой является суммой. Повторяем это действие, пока не получим вероятность вершины 1.

Для данного примера:

$$H = - \sum_x p(x) \log p(x) = 2.4016$$

$$\bar{l} = \sum_x l(x)p(x) = 2.4500$$

5 Подготовка к первой контрольной работе

Мат. ожидание суммы зависимых (и независимых) случайных величин:

$$\begin{aligned}
 E(x + y) &= \sum_i^{|x|} \sum_j^{|y|} (x_i + y_j) p(x_i, y_j) = \\
 &= \sum_i^{|x|} \sum_j^{|y|} (x_i p(x_i, y_j) + y_j p(x_i, y_j)) = \\
 &= \sum_i^{|x|} \sum_j^{|y|} x_i p(x_i) p(y_j | x_i) + \sum_j^{|y|} \sum_i^{|x|} y_j p(x_i) p(y_j | x_i) = \\
 &= \sum_i^{|x|} x_i p(x_i) \sum_j^{|y|} p(y_j | x_i) + \sum_j^{|y|} \sum_i^{|x|} y_j p(y_j) = \\
 &= E(x) + E(y)
 \end{aligned}$$

Мат. ожидание случ. величины и константы:

$$E(c \cdot x) = \sum c \cdot x \cdot p(x) = c \sum x p(x) = c E(x)$$

Мат. ожидание произведения независимых случ. величин:

$$\begin{aligned}
 E(xy) &= \sum_i^{|x|} \sum_j^{|y|} x_i y_j p(x_i) p(y_j) = \\
 &= \sum_i^{|x|} x_i p(x_i) \sum_j^{|y|} y_j p(y_j) = \\
 &= E(x)E(y)
 \end{aligned}$$

Дисперсия двух случайных величин, которые **не коррелируют** ($E[(x - E(x))(y - E(y))] = 0$):

$$\begin{aligned}
 D(x+y) &= \sum_i^{|x|} \sum_j^{|y|} p(x_i) p(y_j) (x_i + y_j - E(x+y))^2 = \\
 &= \sum_i^{|x|} \sum_j^{|y|} p(x_i) p(y_j) (x_i - E(x) + y_j - E(y))^2 = \\
 &= \sum_i^{|x|} \sum_j^{|y|} p(x_i) p(y_j) [(x_i - E(x))^2 + (y_j - E(y))^2 + 2(x_i - E(x))(y_j - E(y))] = \\
 &= \sum_i^{|x|} \sum_j^{|y|} p(x_i) p(y_j) (x_i - E(x))^2 + \sum_j^{|y|} \sum_i^{|x|} p(x_i) p(y_j) (y_j - E(y))^2 + \\
 &\quad + 2 \sum_i^{|x|} \sum_j^{|y|} p(x_i) p(y_j) (x_i - E(x))(y_j - E(y)) = \\
 &= \sum_i^{|x|} p(x_i) (x_i - E(x))^2 + \sum_j^{|y|} p(y_j) (y_j - E(y))^2 + 0 = D(x) + D(y)
 \end{aligned}$$

В более компактной форме:

$$\begin{aligned}
 D(x+y) &= E(x+y - E(x+y))^2 = E(x - E(x) + y - E(y))^2 = \\
 &= E((x - E(x))^2 + (y - E(y))^2 + 2(x - E(x))(y - E(y))) = \\
 &= E(x - E(x))^2 + E(y - E(y))^2 + 2E[(x - E(x))(y - E(y))] = \\
 &= D(x) + D(y)
 \end{aligned}$$

Дисперсия случ. величины и константы:

$$D(cx) = E(cx - E(cx))^2 = E(cx - cE_x)^2 = E(c^2(x - E_x)^2) = c^2 D(x)$$

Удобная формула дисперсии:

$$D(x) = E(x - E(x))^2 = E(x^2 + E^2(x) - 2xE(x)) = E(x)^2 + E^2(x) - 2E(x)E(x) = E(x)^2 - E^2(x)$$

Дисперсия суммы случ. величины и константы:

$$D(x + c) = E(x + c - E(x + c))^2 = E(x + c - E(x) - c)^2 = D(x)$$

По свойству мат. ожидания: $E(c) = c$.

Если $|X| = n$, то $|Y| \leq |X| = 2^n$.

Доказательство через двоичный вектор Пусть подмножество определяется двоичным вектором, каждая компонента которого указывает на принадлежность соответствующего элемента подмножеству. Тогда мощность подмножеств равняется количеству размещений с возвращением: $|\{0, 1\}^n| = 2^n$.

Доказательство через биномиальные коэффициенты Число подмножеств размера k : C_n^k . Тогда для всех размеров: $|Y| = \sum_{i=0}^n C_n^i = C_n^0 + C_n^1 + C_n^2 + \dots$. Такая сумма равна $(1+1)^n = 2^n$.

Можем воспользоваться свойством: $C_{n+1}^k = C_n^k + C_n^{k-1}$. Докажем его:

$$C_n^k + C_n^{k-1} = \frac{n!}{(n-k)!k!} + \frac{n!}{(n-k+1)!(k-1)!} = \frac{n!(n-k+1) + n!k}{(n+1-k)!k!} = \frac{(n+1)!}{(n+1-k)!k!} = C_{n+1}^k.$$

Разложим биномиальные коэффициенты, кроме первого и последнего (у них просто уменьшим n): $C_{n-1}^0 + C_{n-1}^1 + C_{n-1}^0 + C_{n-1}^2 + C_{n-2}^1 + \dots = 2C_{n-1}^0 + 2C_{n-1}^1 + \dots = 2(C_{n-1}^0 + C_{n-1}^1 + \dots) = \dots = 2^n$.

Пусть $X = \{a, b, c\}$,
 $p_1(a) = p_1(b) = p_1(c) = \frac{1}{3}$
 $p_2(a) = p_2(b) = \frac{1}{4}$, $p_2(c) = \frac{1}{2}$.

Найти: $I(x)$, $H(X)$, $I(abaac)$.

$$\begin{aligned} I(a_1) &= I(b_1) = I(c_1) = -\log \frac{1}{3} = \log 3 \\ H_1(x) &= E(I_1) = \log 3 \\ I_1(abaac) &= -\log p(abaac) = -\log(\frac{1}{3})^5 = 5 \log 3 \end{aligned}$$

$$\begin{aligned} I(a_2) &= I(b_2) = \log 4, \quad I(c_2) = \log 2 \\ H_2(x) &= \frac{1}{4} \log 4 + \frac{1}{4} \log 4 + \frac{1}{2} \log 2 = \log 2 + \frac{1}{2} \log 2 \\ I_2(abaac) &= -\log \frac{1}{4^3 \cdot 2} = 9 \end{aligned}$$

Пусть $X = \{a, b, c, d, e, f\}$,
 $P = \{0.4, 0.3, 0.15, 0.05, 0.05, 0.05\}$.

Найти: $H(X)$, \bar{l} .

$$\begin{aligned} H(X) &= -(0.4 \log 0.4 + 0.3 \log 0.3 + 0.15 \log 0.0075) \approx 2.11 \\ \bar{l} &= 0,4 \cdot 1 + 0,3 \cdot 2 + 0,15 \cdot 3 + 0,05 \cdot 4 + 0,05 \cdot 5 + 0,05 \cdot 5 = 0,4 + 0,6 + 0,45 + 0,2 + 0,25 + 0,25 = 2,15 \end{aligned}$$

Избыточность кодирования:

$$R = \bar{l} - H$$

Рассмотрим геометрический источник. Дано множество целых чисел: $I = \{0, 1, 2, 3, \dots\}$.
Вероятность, что источник сгенерирует число из множества: $p(y = i) = (1 - \alpha)\alpha^i$.

Проверим условие нормировки (сумму вероятностей):

$$\sum_{i=0}^{\infty} (1 - \alpha)\alpha^i = (1 - \alpha) \sum_{i=0}^{\infty} \alpha^i = 1 - \alpha \frac{1}{1 - \alpha} = 1$$

Сумму бесконечно убывающей геометрической прогрессии записали по формуле.

$$E(I) = (1 - \alpha) \sum_{i=0}^{\infty} i\alpha^i.$$

Общий член ряда похож на производную степени:

$$\left(\sum_{i=0}^{\infty} \alpha^i\right)' = \sum_{i=0}^{\infty} i\alpha^{i-1} = \sum_{i=0}^{\infty} \frac{i\alpha^i}{\alpha}.$$

Выразим ряд через производную:

$$E(I) = (1 - \alpha) \left(\sum_{i=0}^{\infty} \alpha^i \right)' \alpha = (1 - \alpha) \left(\frac{1}{1 - \alpha} \right)' \alpha = \frac{\alpha}{1 - \alpha}$$

Найдем дисперсию:

$$D(X) = E(X^2) - E^2(X).$$

Найдем первую часть:

$$E(X^2) = (1 - \alpha) \sum_{i=0}^{\infty} i^2 \alpha^i$$

Выразим ряд через производную:

$$\left(\sum i\alpha^i\right)' = \sum i^2 \alpha^{i-1} = \frac{\sum i^2 \alpha^i}{\alpha}$$

$$E(X^2) = (1 - \alpha) \left(\sum i\alpha^i \right) \alpha = (1 - \alpha) \left(\left(\sum a^i \right)' \alpha \right)' \alpha = (1 - \alpha) \left(\frac{\alpha}{(1 - \alpha)^2} \right)' \alpha = \frac{\alpha(1 + \alpha)}{(1 - \alpha)^2}$$

Таким образом, дисперсия:

$$D(X) = \frac{\alpha(1 + \alpha)}{(1 - \alpha)^2} - \frac{\alpha^2}{(1 - \alpha)^2} = \frac{\alpha}{(1 - \alpha)^2}$$

Рассмотрим кодирование длин серий:

Пусть $X = \{0, 1\}$. Источник определяется последовательностью символов: 0011000010100. Перейдем к новому источнику, для этого перед каждой единицей запишем количество нулей: $0^2 1 0^0 1 0^4 1 0^1 1 0^2 \rightarrow 2, 0, 4, 1, 2$. Будем кодировать не символы, а полученные длины серий, то есть $X \rightarrow I = 0, 1, 2, 3 \dots$.

Если $p(0) = \alpha$, то вероятность, что длина серии равна i : $p(y = i) = (1 - \alpha)\alpha^i$

Найдем энтропию источника (ансамбля $\{p(i), i \in I\}$):

$$\begin{aligned} H &= - \sum_i (1 - \alpha)\alpha^i \log[(1 - \alpha)\alpha^i] = \\ &= -(1 - \alpha) \log(1 - \alpha) \sum_i \alpha^i - (1 - \alpha) \log \alpha \sum_i i \alpha^i = \\ &= -\log(1 - \alpha) - \frac{\alpha \log \alpha}{1 - \alpha} = \frac{-(1 - \alpha) \log(1 - \alpha) - \alpha \log \alpha}{1 - \alpha} = \frac{\eta(\alpha)}{1 - \alpha} \end{aligned}$$

$$X = \{1, 2, 3, \dots, M\}, p(1) = p(2) = \dots = p(M) = \frac{1}{M}$$

Если будем кодировать кодом Хафмана, а $M = 2^n$, то длина кода любого символа $\log_2 M = n$, значит $H(X) = \log M$, $\bar{l} = \log M$, значит получаем оптимальный код. Если количество символов не является степенью двойки, то код будет не оптимальным.

Пусть $\lfloor \log M \rfloor = S$ — длина кодового слова в полном дереве, $M - 2^S = F$ — количество веток, которые добавили к полному дереву. Тогда средняя длина кодового слова при равных вероятностях

$$\frac{S(2^S - F) + (S + 1)2F}{M}$$

Рассмотрим марковскую цепь:

$$A = \begin{bmatrix} 1/2 & 1/2 \\ 1/3 & 2/3 \end{bmatrix}$$

Пусть начальное распределение: $p = \{3/4, 1/4\}$.
 $I(110) = -\log 1/4 \cdot 2/3 \cdot 1/3$

6 Канальное кодирование

После сжатия информации, нам нужно ее передать. При этом сохранить ее достоверность и целостность (без потерь).

Начало курса по теории кодирования. Будем рассматривать кодер (и декодер) канала.

Кодер канала обеспечивает заданную достоверность при передаче или хранении информации путем дополнительного внесения избыточности.

Достоверность не обеспечивается на 100 процентов. Какие-то потери будут. Цель кодера канала — обработать информацию для защиты при передаче.

Изучаем помехоустойчивое кодирование. Нужно ошибку обнаружить, затем исправить. Под ошибкой понимается нарушение целостности данных.

Для обнаружения ошибок используют коды обнаружения ошибок, а для исправления корректирующие коды (корректирующие коды также могут использоватьсь для обнаруживания).

Стратегии борьбы с ошибками в системах связи:

- обнаружение ошибки в блоке данных и автоматический запрос повторной передачи поврежденного блока,
- обнаружение ошибки и отбрасывание поврежденного блока (хорошо применяется в системах потоковой передачи),
- исправление ошибки на физическом уровне.

Критерии кодов, исправляющих ошибки: однозначность, минимальная избыточность, устойчивость к ошибкам.

Однозначность — получение после декодирования в точности то, что кодировали.

Под устойчивостью к ошибкам имеется ввиду исправление ошибок, возникших в процессе передачи.

Виды ошибок:

- ошибка замещения: муха \rightarrow мука,
- ошибка стирания: муха \rightarrow муха (легче заметить, чем ошибку замещения, так как уменьшается размер данных),
- ошибка выпадания: муха \rightarrow уха (просто выпала буква)
- ошибка вставки: мука \rightarrow мурка
- комбинационная ошибка

Типы ошибок:

- детерминированный (в канале не более n символов может быть изменено),
- вероятностный (каждый символ с вероятностью p меняется на другой символ)

Пусть A_q алфавит канала, $|A_q| = q$ (q — бит, 0 или 1). Т.е. двоичный алфавит. n — длина кода (каждое слово имеет длину n бит). Мощность кода $|C|$ — количество кодовых слов, которое можем получить.

Вес слова $\|a\|$ — количество ненулевых элементов в слове ($\{i : a_i \neq 0\}$).

Пусть a и b — слова в алфавите канала. $\tilde{d}(a, b)$ — минимальное число ошибок в результате которых a переходит в b (кодовое расстояние). Так как b — разрешенное слово, то декодер не сможет распознать ошибку. Задача обнаружить d .

Способ кодирования позволяет исправить k ошибок, если при передаче в канал различных кодовых сообщений, на выходе будут получаться различные сообщения:

$$\exists a', a'' \in C, a : (a' \neq a'' \wedge \tilde{d}(a', a) \leq k \wedge \tilde{d}(a'', a) \leq k)$$

Кодовое расстояние кода C :

$$\tilde{d}(C) := \min_{a \neq b, a, b \in C} \tilde{d}(a, b)$$

Кодовое расстояние определяет устойчивость к ошибкам. C обнаруживает t ошибок, значит $\tilde{d}(C) > t$, C исправляет t ошибок, значит $\tilde{d}(C) > 2t$.

Блочный код — тип канального кодирования, исходное сообщение делится на блоки, которые преобразуются для исправления ошибок. Системе блочного кодирования на вход подается k -значное кодовое слово W , на выходе получается n -значное кодовое слово $C(W)$, которое называется блоком.

Линейное кодирование. Пусть $p \in P$, $m \in N$, $q = p^m$, $C \subseteq F_q^n$. Код C называется линейным, если он является линейным подпространством F_q^n . n — длина слов, M — количество, d — расстояние. Обозначается $(n, M, d)_q$.

сообщение кодовое слово Источник сообщений -> кодер -> Канал в канал попадает шум

Линейный код — тип блочного кодирования. Плюсы: работает быстро и легко. Минусы: легко взламывается, так как линейный. В кодере происходит линейная операция, поэтому код называется линейным.

Основные понятия:

- Сообщение — k символов,
- К k символам добавляем избыточность (проверочные символы) ($n - k$ символов)
- кодовое слово — n символов
- Код — множество допустимых слов, которые можем получить

Код с проверкой на четность. В конец каждого блока добавляется бит четности. Декодер проверяет четность блока и использует этот бит. Если произошла ошибка, блок запрашивается повторно. Такой код не является корректирующим, только детектирует ошибку.

$$C := \{(a_1, \dots, a_n) \in F_2^n \mid \sum a_i = 0\}$$

Слово называется четным, когда мы складываем все его 1 и получаем 0. Значение добавочного бита (a_n) такое, чтобы слово было четным. Один проверочный бит и $n - 1$ информационных битов.

7 Коды Хемминга

Код — это множество допустимых кодовых слов.

Проверочная матрица размерности $n - k$ (длина избыточности) на n (длина слова)

ва):

$$H \cdot x^T = 0$$

x — кодовое слово. Если имеем 0, то кодовое слово называется легитимным (не ошибочным).

Проверочная матрица состоит из

$$H = (A \mid I_{n-k})$$

A определяется кодом, описывается самим алгоритмом.

Определение. Пусть H — любая двоичная матрица. Тогда линейный код с проверочной матрицей H состоит из всех векторов x таких, что $Hx^T = 0$.

Если H имеет $n-k$ линейно независимых строк, то имеется 2^k кодовых слов, это называется размерностью кода.

Код называется $[n, k]$ кодом.

Код использует n символов для передачи k символов, эффективность или скорость кода: $R = k/n$.

Если есть сообщение, то как найти соответствующее кодовое слово? Порождающая матрица:

$$G = [I_k \mid -A^T]$$

Конкатенация единичной и транспонированной.

Формула в общем виде. В двоичном пространстве $a = -a$.

Картинка

Нужно найти и убрать вектор ошибок. Убрать легко, найти сложно.

Расстояние Хэмминга — количество позиций, где слова отличаются.

Вес Хэмминга — количество ненулевых символов символов в строке (последовательности)

Связь веса с расстоянием:

$$d(a, b) = wt(a - b)$$

$$d = \min_{d(a,b)} = \min_{wt(a-b)}$$

Теорема 1. Минимальное расстояние линейного кода равно минимальному весу ненулевых слов.

Теорема 2. Код с минимальным расстоянием может исправлять наибольшее целое число от $d - 1/2$ ошибок.

Пусть $a \in C$, $a \rightarrow b$, $e = b - a$ — вектор ошибок.

$$b = a + e$$

$$H \cdot b^T = H \cdot a^T + H \cdot e^T = H \cdot e^T$$

$$\|e\| \leq t, \quad e!$$

Ошибок может быть множество, t — количество ошибок при переходе, векторов ошибок не больше t . $H \cdot a^T = 0$ по определению.

Синдром принятого слова b :

$$H \cdot e^T = s$$

$$s = H \cdot b^T$$

Если синдром равен нулю, то вектор B не болен и в нем нет ошибки)

Смежный класс (сдвиг) вектора a из кода C :

$$a + c = \{a + x, x \in C\}$$

Плюсы везде означают XOR (плюс в круге).

Два вектора лежат в одном смежном классе, когда $a - b \in C$.

Как получить кодовое слово из сообщения:

$$c = xG$$

Линейный $[n, k]$ код. Строки порождающей матрицы линейно независимы.

Стандартное расположение — таблица, состоящая из: Сам код с нулевым кодовым словом. Смежные классы (лидер класса с левой стороны) (просто нулевой вектор с одной единицей, который последовательно ксорится с кодами)

Вставить тут таблицу

Нет смежного класса с лидером 0001, так как это значение уже есть во втором смежном классе.

Смысл смежных классов в том, что мы получаем все выходные вектора с ошибкой в соответствующем бите.

Смотрим пришедший вектор, иксорим его с лидером смежного класса и получаем код без ошибки.

Теорема 3. Для двоичного кода синдром равен сумме тех столбцов, где произошла ошибка.

Теорема 4. Синдром равен 0 только если b является исходным кодовым словом.

Если синдром перевести в число, то это будет индекс столбца, где произошла ошибка (на картинке перепутаны направления).

Двоичный код Хэмминга длины $n = 2^r - 1$ имеет проверочную матрицу H , столбцы которой состоят из всех ненулевых двоичных векторов длины r , каждый вектор встречается один раз. $n = 2^r - 1$, $k = 2^r - 1 - r$, $d = 3$. Код хэмминга — это двоичное представление чисел от 0 до n .

Не всегда проверочная матрица представляется в канонической форме, когда четко видно единичную матрицу. Путем элементарных преобразований, матрицу Хэмминга можно привести к канонической.

Линейный код длины n называется циклическим, если для любого кодового слова (x_1, x_2, \dots, x_n) слово $(x_2, x_3, \dots, x_n, x_1)$ также является кодовым.

8 Коды Варшамова-Тенегольца

В основном исправляются ошибки замещения.

Особенности:

- Ошибки выпадения
- Ошибки вставки
- Простой алгоритм исправления ошибок

Ошибки замещения сложней детектировать, так как не изменяется размер данных.

Определение кодов:

$$C := \{a_1, \dots, a_n \mid \sum_{i=1}^n ia_i = 0 \pmod{n+1}\}$$

Кодовым пространством являются слова, где сумма по модулю равна нулю.

Пример: $n = 6$, 000011 , $(5 + 6) \neq 0 \pmod{7}$ (кодовое слово не принадлежит коду Варшамова-Тенегольца)

$$n = 6, 100001, (1 + 6) = 0 \pmod{7}$$

Опять биты слева на право?

Рассмотрим ошибку выпадения.

На входе: $a = a_1, \dots, a_n$. На выходе: $a' = a_1, \dots, a_{k-1}, a_{k+1}, a_n$.

Нужно восстановить a_k и a по a' .

Пример: $a = 010001$, $a_k = 0$, $k = 4$
 $a' = 01001/010001$

Восстановить $a \neq (a_k, k)$. То есть нам не нужно восстанавливать позицию выпадения, достаточно восстановить диапазон.

Пусть $n_0 = \{i > k \mid a_i = 0\}$, $n_1 = \{i > k \mid a_i = 1\}$ (количество нулей или единиц справа от k элемента).

Если $a_k = 0$, то a можно восстановить по a' , если известно n_1 . Пример: $n_1 = 3, \dots 010101$. Идем справа налево. Когда дошли до 3-й единицы, сразу вставили 0. Для $a_k = 1$ аналогично.

$$S := \sum_{i=1}^n ia_i, \quad S' := \sum_{i=1}^{n-1} ia'_i$$

, где S' — сумма кода, полученного из канала., $n - 1$, т.к. один элемент выпал

Вторую сумму можем посчитать, первую не знаем.

Заметим, что

$$S - S' = \sum_{i=1}^n ia_i - \left(\sum_{i=1}^{k-1} ia_i + \sum_{i=k}^{n-1} ia_{i+1} \right) = \sum_{i=k}^n ia_i - \sum_{i=k+1}^n (i-1)a_i = ka_k + \sum_{i=k+1}^n a_i$$

Там $i + 1$ индекс чтобы перейти от a' к a .

Выразим отсюда S' , получим

$$S' = S - \left(ka_k + \sum_{i=k+1}^n a_i \right) = S - ka_k - n_1$$

Так как $S = 0 \bmod n + 1$, то

$$S' = -n_1 - ka_k \bmod n + 1$$

Если $a_k = 0$, то $-S' = n_1$.

Если $a_k = 1$, то $-S' = n_1 + k = (n - k - n_0) + k = n - n_0$

$$a_k = 0 \implies -S' \bmod n + 1 = n_1$$

$$a_k = 1 \implies -S' \bmod n + 1 = n - n_0$$

Заметим, что $\|a'\| \geq n_1$ (количество единиц в кодовом слове больше либо равно) и $\|a'\| \leq (n - 1) - n_0$, отсюда $n_1 \leq \|a'\| < n - n_0$.

Алгоритм нахождения выпавшего символа:

Вычисляем $-S' \bmod n + 1$

Сравниваем $-S' \bmod n + 1$ с $\|a'\|$.

Если $\|a'\| \geq -S' \bmod n + 1$, то выпал 0, а если $\|a'\| \leq n - (-S') \bmod n + 1$, то

выпала 1.

Пример: $a' = 10001$ восстановить a .

8.1 Ошибка вставки

На входе: $a = a_1, \dots, a_n$

На выходе: $a' = a'_1, \dots, a'_n = a_1, \dots, a_k, x, a_{k+1}, a_n$

Рассмотрим частные случаи вставки в начало и конец. Если $k = 0$, то $a' = xa$, если $k = n + 1$, то $a' = ax$.

$$S' = S + (k + 1)x + \sum_{i>k} a_i$$

$$\begin{aligned} S' \mod n + 1 &= 0 + (k + 1)x \mod n + 1 + \sum_{i=k} a_i \mod n + 1 = \\ &= (k + 1)x + \sum_{i>k} a_i = (k + 1)x + n_1 \end{aligned}$$

Ведем обозначение: $T := S' \mod n + 1$

Выделим 3 случая:

- $T = 0 \implies$ удаляем из a' последний символ и получаем a ,
- $T = ||a'|| \implies$ удаляем первый символ,
- $0 < T \neq ||a'||$. Если $x = 0$, тогда $T < ||a'||$, $T = n_1$
Если $x = 1$, тогда $T > ||a'||$ $T = n + 1 - n_0$

Первый случай. Если $T = 0$, то $(k + 1)x + n_1 = 0$, либо $(k + 1)x + n_1 = (n + 1)z$, где $z = 1, \dots$.

9 Кодирование дискретных источников при неизвестной статистике

Дано 4 символа:

$$X = \{A, B, C, D\}.$$

Энтропия определенного ансамбля:

$$H = -p_A \log p_A - p_B \log p_B - p_C \log p_C - p_D \log p_D.$$

Представим символы в двоичном виде:

$$X = \{00, 01, 10, 11\}.$$

Затраты на первый разряд (считываются справа налево):

$$H_0 = -(p_A + p_C) \log(p_A + p_C) - (p_B + p_D) \log(p_B + p_D).$$

Мы рассматриваем два случая: когда в разряде ноль и когда там единица. Затраты на второй разряд:

$$H_1 = -(p_A + p_B) \log(p_A + p_B) - (p_C + p_D) \log(p_C + p_D).$$

Затраты на второй разряд, если знаем первый:

$$H_{1|0} = -\left(\frac{p_A}{p_A + p_C} + \frac{p_B}{p_B + p_D}\right) \log\left(\frac{p_A}{p_A + p_C} + \frac{p_B}{p_B + p_D}\right) - \left(\frac{p_C}{p_A + p_C} + \frac{p_D}{p_B + p_D}\right) \log\left(\frac{p_C}{p_A + p_C} + \frac{p_D}{p_B + p_D}\right)$$

Здесь в первом случае во втором разряде ноль, но в первом разряде может на-

ходиться либо ноль, либо единица (вспомнить формулу условной вероятности).

Имеем независимые ансамбли, поэтому суммарные затраты:

$$H = H_0 + H_1 = -p_A(\log(p_A + p_C) + \log(p_A + p_B)) \dots$$

В данном случае энтропия больше, чем энтропия символов.

Рассмотрим унарную бинаризацию. Будем бинаризовать символы следующим образом (передача слева направо):

$$A = 1, B = 01, C = 001, D = 0001$$

Если после начала передачи получили 1, то понимаем, что это А. Если получили 0, то ждем следующий символ. Если им будет 1, то понимаем, что это В.

Четвертый символ может быть только 1:

$$H_3 = 0$$

Первый символ может быть 1 или 0:

$$H_0 = -p_A \log -(1 - p_A) \log$$

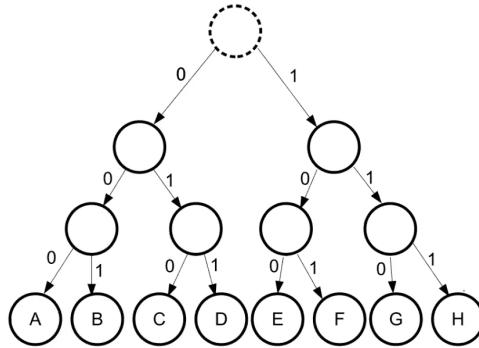
Второй символ может быть 1 или 0, при условии, что первый был 0:

$$H_1 = -\frac{p_B}{1-p_A} \log -\frac{p_C+p_D}{1-p_A} \log$$

Третий символ по аналогии со вторым (в знаменателе будет $p_C + p_D$).

$$H = H_1 + (p_B + p_C + p_D)H_2 + (p_C + p_D)H_3 + p_DH_4 = H \dots$$

Третий способ бинаризации — двоичное арифметическое кодирование (древовидная бинаризация):



Затраты на первый символ (будет 0 или 1):

$$H_1 = -(p_A + p_B) \log -(p_C + p_D) \log$$

Затраты на второй символ (учитываем вероятность перехода в соответствующее поддерево):

$$H_2 = -(p_A + p_B)\left(\frac{p_A}{p_A + p_B} \log + \frac{p_B}{p_A + p_B} \log\right) - (p_C + p_D)\left(\frac{p_C}{p_C + p_D} \log + \frac{p_D}{p_C + p_D} \log\right)$$

Если это упростить, то получится: $H_2 = -p_A \log \frac{p_A}{p_A + p_B} - p_B \log \frac{p_B}{p_A + p_B} - p_C \log \frac{p_C}{p_C + p_D} - p_D \log \frac{p_D}{p_C + p_D}$.

Теперь распишем логарифм частного и упростим еще раз: $H_2 = H + (p_A + p_B) \log(p_A + p_B) + (p_C + p_D) \log(p_C + p_D)$

$$H_2 = H - H_1$$

Общие затраты:

$$H_1 + H_2 = H$$

Данный способ бинаризации при переходе от недвоичного кодера к двоичному не меняет эффективность кодирования.

Унарная и древовидная бинаризация дают ту же энтропию, которая была без бинаризации.

Допусим, есть недвоичный источник с памятью 1. Хотим его кодировать двоичным арифметическим кодером.

Алгоритмы кодирования, такие как код Хаффмана или арифметическое кодирование, требуют знания распределения вероятностей символов источника. В реальных задачах это распределение не известно.

Проще говоря, нам не известны вероятности символов.

Можно выделить два способа решения этой проблемы. Первый заключается в подсчете частоты каждого символа в файле источника. Так как декодеру не доступен исходный файл, нам нужно передать частоты вместе с сообщением. Этот способ называется двухпроходным кодированием или off-line coding.

Второй способ заключается в оценке вероятности каждого символа исходя из предыдущих символов. Оценка осуществляется одинаково на стороне кодера и декодера. Способ называется адаптивным кодированием или online coding.

Постановка задачи Кодируем стационарные источники (распределение вероятностей не меняется). Пусть нам известны вероятности, тогда вычислим энтропию H , которая является нижней границей средней скорости кодирования R . Метод универсального кодирования хороший, если он обеспечивает R , близкую к той, которая была получена при известных вероятностях.

$\Omega = \{w\}$ — множество моделей источников. Если Ω — множество дискретных постоянных источников, то каждая w определяется распределением вероятностей $(w) = (\theta_1, \dots, \theta_k)$. w и θ не известны ни кодеру, ни декодеру.

$H(w)$ — энтропия для заданной модели.

$\bar{R}_n(w)$ — средняя по последовательности из n символов скорость кодирования для w .

$r_n(w) = \bar{R}_n(w) - H(w)$ — средняя избыточность для модели w и длины последовательности n .

Задача заключается в построении алгоритма:

$$r_n(\Omega) = \max_{w \in \Omega} r_n(w)$$

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

жества Ω .

Двухпроходное кодирование от адаптивного отличается тем, что в первом случае имеется большая задержка, потоковую передачу реализовать не получится.

9.1 Двухпроходное кодирование

Проход 1: оценить θ (оценка $\bar{\theta}$), кодировать $\bar{\theta}$ (кодовые слова c_1).

Проход 2: кодировать символы источника x используя $\bar{\theta}$ (коды c_2), сформировать кодовое слово из двух частей $c = (c_1, c_2)$.

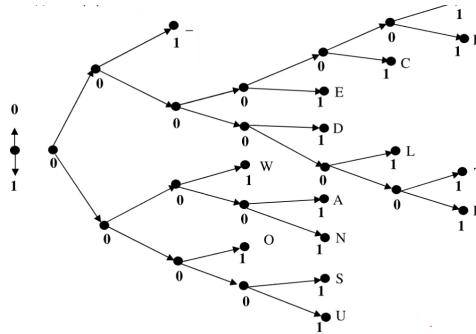
Рассмотрим двухпроходное кодирование на примере кода Хаффмана. Пусть есть источник, который генерирует буквы. Всего букв 256 (8 бит на букву). Передается сообщение «IF WE CANNOT DO AS WE WOULD WE SHOULD DO AS WE CAN». В нем 50 символов. При равномерном кодировании затратим $50 \times 8 = 400$ бит.

Вычислим частоту каждой буквы в сообщении и закодируем каждую букву кодом 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

Последний столбец содержит количество бит, которое будет затрачено на букву в сообщении.

Мы получили только вторую часть кодового слова (c_2). Чтобы декодер смог ее обработать, передадим ему полученное ранее дерево Хаффмана.



$$c_1 = (0\ 00\ 1000\ 001010\ 01101111\ 0110\ 1111, \text{ASCII}(x), \dots)$$

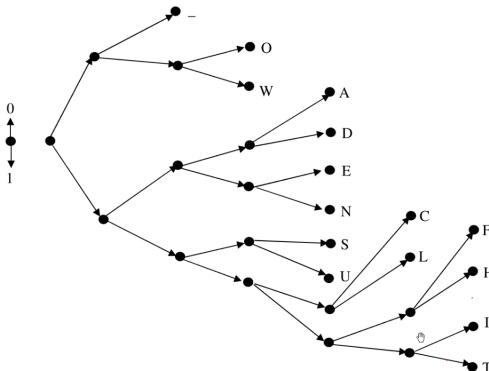
$$l_1 = 29 + 8 \times 15 = 149 \text{ бит}, l = l_1 + l_2 = 149 + 178 = 327 \text{ бит}$$

Будем кодировать каждый уровень дерева сверху вниз, 0 обозначим неконцевую

вершину, 1 — концевую. На это будет затрачено 29 бит. Чтобы связать концевые вершины с буквами, включим их в виде ASCII кодов (15 букв по 8 бит).

Метод можно улучшить, если строить канонический код Хаффмана. Если вероятности нескольких символов совпадают, то можно по одному распределению построить несколько кодов. Средняя длина этих кодов будет одинаковой, но если короткие кодовые слова будут лексикографически предшествовать более длинным, то на передачу дерева будет затрачено меньшее количество бит.

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



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

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

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

Для каждого уровня будем использовать количество бит, необходимое для записи числа вершин на этом уровне. Используя доступные биты для каждого уровня запишем число концевых вершин.

Оценим избыточность.

Теорема Полное кодовое дерево, имеющее концевых вершин, имеет $M - 1$ промежуточных (внутренних) вершин. Поэтому $M + M - 1 = 2M - 1$ бит достаточно для описания дерева.

$$l_1(x) \leq 2M - 1 + M \lceil \log M \rceil$$

$$l_2(x) = \sum_{i=1}^n l(x_i) = \sum_{x \in X} \tau(x) l(x)$$

Внесем n в числитель и знаменатель и получим оценку вероятности символа:

$$\begin{aligned} n \sum_{x \in X} \frac{\tau(x)}{n} l(x) \\ n \sum_{x \in X} \hat{\theta}(x) l(x) = n E_{\hat{\theta}}\{I(x)\} \leq n(H(\hat{\theta}_n) + 1) \end{aligned}$$

Вычислим среднее по всем возможным вариантам n : $E(H(\hat{\theta}_n)) \leq H(E(\hat{\theta}_n)) = H(\theta) = H$

$$\bar{R}(x) = \frac{l_1(x) + l_2(x)}{n} \leq H + 1 + \frac{2M - 1 + M \lceil \log M \rceil}{n}$$

9.2 Нумерационное кодирование

Последовательность на выходе источника $\mathbf{x} \in X^n$, $X = \{0, 1, \dots, M-1\}$. Стока из n символов алфавита длины M .

Композиция $\tau(\mathbf{x}) = (\tau_0(\mathbf{x}), \dots, \tau_{M-1}(\mathbf{x}))$, $M = |X|$. Вектор, состоящий из количества каждой буквы словаря в сообщении.

Кодирование:

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

Выделим 3 способа кодирования композиции. Первый заключается в кодировании каждого $\tau_i(\mathbf{x})$, кроме $i = M-1$, прямым кодом, используя $\lceil \log(n+1) \rceil$ бит. В данном случае нам важно знать количество букв в сообщении n . Количество последнего символа можем вычислить по предыдущим значениям, поэтому его можно не передавать. Недостаток способа в большом количестве бит.

Второй способ заключается в представлении композиции в виде двоичной последовательности вида $0^{\tau_0} 1^{\tau_1} \dots, 1^{M-1}$, которая имеет длину $n + M - 1$ и вес (количество единиц) $M - 1$. В последовательности n нулей, единицы используются в качестве разделителя.

В данном случае имеем $N_\tau(n, M) = \binom{n+M-1}{M-1}$ строк такой же длины и веса. Упорядочим их лексикографически и закодируем равномерным кодом номер последовательности, используя $\lceil N_\tau(n, M) \rceil$ бит. Недостаток способа в сложности.

Третий способ заключается в упорядочивании Q ненулевых компонент τ по убыванию: $\tau_0 \in \{1, \dots, n\}$, $\tau_1 \in \{1, \dots, \tau_0\}$, $\tau_2 \in \{1, \dots, \tau_1\}, \dots$.

Упорядоченную композицию τ_Q кодируем арифметическим кодером (АК) с вероятностями $\frac{1}{n}, \frac{1}{\tau_0}, \frac{1}{\tau_1}, \dots, \frac{1}{\tau_Q-2}$. Смысл упорядочивания в том, что вероятности увеличиваются, так как сужается диапазон, в котором может находиться количество следующего символа. После сортировки композиции, порядок ее компонентов не соответствует порядку букв в словаре. При помощи АК кодируем буквы, которые соответствуют компонентам композиции с вероятностями: $\frac{1}{M}, \frac{1}{M-1}, \frac{1}{M-2}, \dots, \frac{1}{M-Q+2}$.

Возьмем сообщение: IF WE CANNOT DO AS WE WOULD WE SHOULD DO AS WE CAN. Здесь $n = 50$, $M = 256$, $Q = 15$.

Для первого способа:

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

Для второго способа:

$$l_1 = \lceil \log \left(\frac{255 + 50}{255} \right) \rceil = 193 \text{ бит.}$$

Для третьего способа:

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

$$l_1 = \lceil \log(50 \cdot 12 \cdot 5^2 \cdot 4^3 \cdot 3^2 \cdot 2^3 \cdot 256 \cdot 255 \cdots 243) \rceil = 27 + 120 = 147 \text{ бит.}$$

$$-\log \frac{1}{50} - \log \frac{1}{12} - \cdots = \log(50 \cdot 12 \cdots)$$

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

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

Среди всех символов выбираем τ_0 символов, которые будут 0, затем среди оставшихся $n - \tau_0$ выбираем τ_1 символ, которые будут 1. Остальные символы будут 3 (один случай).

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

В общем случае для алфавита объемом M имеем:

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

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

$$l = l_1 + l_2 = 147 + 150 = 297$$

Кодирование арифметическим кодером:

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
...

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

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

Избыточность (применим аппроксимацию Стирлинга)

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

9.3 Адаптивное кодирование

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 .

В основном для двухпроходного кодирования используется код Хаффмана, а для адаптивного кодирования — арифметическое. Хотя ничего не мешает использовать код Хаффмана и для адаптивного кодирования. В этом случае кодеру и декодеру синхронно придется перестраивать дерево после каждого символа (или каждые несколько символов), так как меняются вероятности. В отличие от двухпроходного, в адаптивном нам не нужно передавать дерево, это плюс.

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

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

Не можем оценить вероятность символа как 0, так как обнуляется арифметический кодер.

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

$$\text{Более точная поправка: } \hat{p}_t(a) = \frac{\tau_t(a)+1/2}{t+M/2}$$

IF WE CANNOT DO AS WE WOULD WE SHOULD DO AS WE CAN			
$\tau = (12, 5, 5, 4, 4, 4, 3, 3, 2, 2, 2, 1, 1, 1, 0, \dots)$			
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		$(2+1)/(49+256)=3/305$	$(2*2+1)/(2*49+256)=5/354$

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

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

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

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

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

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

Можно использовать подход, основанный на escape-символе. Добавим дополнительный символ в алфавит. Этот символ передается, если на вход приходит символ, который ранее не появлялся.

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

Передается "esc" если $\tau_t(a) = 0$, $p_t(\text{esc}) = \frac{1}{t+1}$.

Алгоритм A:

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

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

Алгоритм D:

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

Сравним

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	$1*1/256$	$1/256$
1	F	$1/2*1/255$	$1/2*1/255$
2		$1/3*1/254$	$2/4*1/254$
3	W	$1/4*1/253$	$3/6*1/253$
4	E	$1/5*1/252$	$4/8*1/252$
5		$1/6$	$1/10$
...
12		$2/13$	$3/24$
...
15		$3/16$	$5/30$
...
49	N	$2/50$	$3/98$

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

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

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

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

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

где

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

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

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

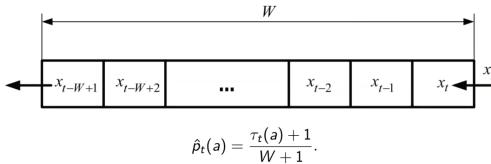
где K не зависит от длины последовательности n .

Сравним

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

Чтобы оценить эффективность алгоритма, можно посчитать энтропию при условии что нет зависимости между символами (энтропия нулевого порядка), либо считать условную энтропию с учетом одного или более предыдущих символов. Либо взять архиватор, скать им, посмотреть, как получится.

Кроме алгоритма А или D можно считать оценку вероятности при помощи скользящего окна. Будем считать количество символов в окне.



В реальных приложениях используется двоичный кодер. Поэтому рассмотрим случай, когда в окне только 0 и 1. Для подсчета этих символов достаточно одного счетчика.

Будем аппроксимировать окно. 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. Из окна удаляется среднее число единиц:

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

Новый символ добавляется в окно:

$$s_{t+1} + x_t \rightarrow s_{t+1}$$

Итоговое правило обновления

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

10 Сжатие изображений 1

Под сигналом понимается физический процесс (например, изменяющееся во времени напряжение), отображающий некоторую информацию.

Аналоговые сигналы — это непрерывные функции непрерывного аргумента (вре-

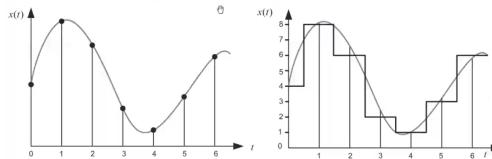
мя).

Дискретные сигналы могут быть дискретными по значению функции или по аргументу.

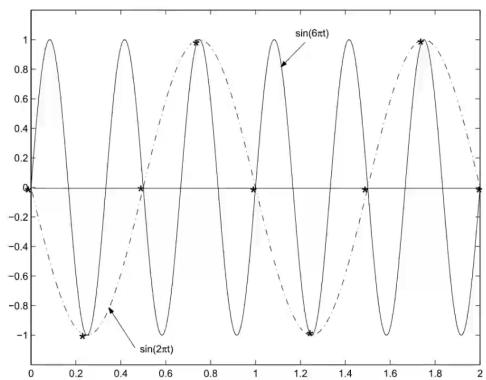
Цифровой сигнал дискретизирован и по уровню, и по времени. Чтобы аналоговый сигнал сохранить на носитель, его преобразовывают в цифровой при помощи аналогово-цифрового преобразователя (ЦАП).

ЦАП включает 2 операции: квантование по времени и квантование по уровню.

Как выбрать шаг квантования? На сколько дискретных уровней разбивать сигнал?



Если выбрать слишком низкую частоту дискретизации (по времени), то восстановленный аналоговый сигнал может быть искажен:



Пусть $x(t)$ — непрерывная одномерная функция. Преобразование Фурье (спектр) для $x(t)$ определяется:

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt,$$

где $w = 2\pi f$ — круговая частота. Эта комплексная функция может быть представлена в виде

$$X(f) = A(f)e^{j\phi(f)},$$

, где $|A(f)|$ называется амплитудным спектром сигнала или амплитудно-частотной характеристикой сигнала, а $\phi(f)$ — фазовым спектром или фазово-частотной характеристикой.

Восстановить сигнал по его спектру можно при помощи обратного преобразования Фурье:

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{-j\omega t} df.$$

Теорема Котельникова-Найквиста Пусть задана функция $x(t)$ и ее спектр имеет вид

$$\begin{cases} X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi f t} dt, & |f| \leq f_0 \\ X(f) = 0, & |f| > f_0 \end{cases}$$

Тогда эта функция полностью определяется своими мгновенными значениями в моменты, отстоящие друг от друга на $\frac{1}{2f_0}$ секунд:

$$x(t) = \sum_k \left(\frac{k}{2f_0} \right) \frac{\sin \pi(2f_0 t - k)}{\pi(2f_0 t - k)}$$

То есть, частота дискретизации должна быть хотя бы в 2 раза больше максимальной частоты сигнала.

С изображением все аналогично, только теперь сигнал двумерный. Спектр изображения, определяемый функцией яркости $l(x, y)$:

$$L(w_x, w_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} l(x, y) e^{-j(w_x x + w_y y)} dx dy.$$

Спектральная интенсивность изображения:

$$S(w_x, w_y) = \frac{1}{x_0 y_0} |L(w_x, w_y)|^2$$

, где $x_0 y_0$ — площадь прямоугольника, в которое вписано изображение.

Восстановить функцию яркости по его спектру можно при помощи обратного двумерного преобразования Фурье:

$$l(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} L(w_x, w_y) e^{j(w_x x + w_y y)} dw_x dw_y.$$

Изображение у нас черно-белое, поэтому определяется одной функцией яркости.

Спектр для $l(x, y)$ по теореме Котельникова:

$$L(w_x, w_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} l(x, y) e^{-j(2\pi f_x x + 2\pi f_y y)} dx dy,$$

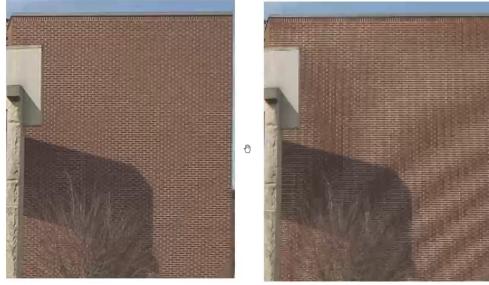
$$|f_x| \leq f_x^0, |f_y| \leq f_y^0$$

$$L(w_x, w_y) = 0,$$

Тогда эта функция полностью определяется при отставании на интервалы $\Delta_x = \frac{1}{2f_x^0}$ и $\Delta_y = \frac{1}{2f_y^0}$ в направлении осей x и y :

$$l(x, y) = \sum_n \sum_k L(n\Delta_x, k\Delta_y) \frac{\sin 2\pi f_x^0(x - n\Delta_x)}{2\pi f_{0x}(x - n\Delta_x)} \cdot \frac{\sin 2\pi f_y^0(y - k\Delta_y)}{2\pi f_{0y}(y - k\Delta_y)}$$

Пример нарушения теоремы (aliasing):



Наиболее прострой способ аппроксимации последовательности сообщений $x = \{x_1, \dots, x_n\}$ может быть реализован при помощи процедуры равномерного скалярного квантования, которая каждому символу x_i сопоставляет номер кванта

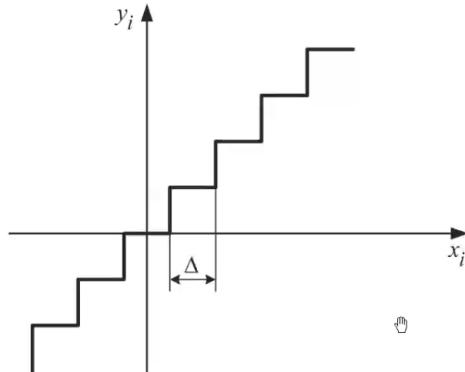
$$z_i = (x_i) \lfloor \frac{|x_i| + \Delta/2}{\Delta} \rfloor,$$

где Δ — шаг квантования

При этом, аппроксимирующее множество $y = \{y_1, \dots, y_n\}$ вычисляется как

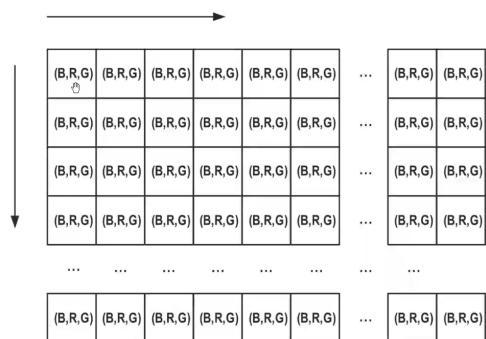
$$y_i = \Delta \cdot z_i.$$

Функция аппроксимации:



Средняя ошибка квантования $\epsilon = \int_{-\infty}^{\infty} (x_i - y_i)^2 f(x_i) dx \approx \frac{\Delta^2}{12}$.

Формат хранения изображений RGB24:



По байту на яркость каждого из 3 цветов: 24 бита на пиксель.

Стрелками показано направление обхода.

Здесь для каждой компоненты используется равномерный код (1 байт). Задача сжать изображение так, чтобы постратить меньше чем 24 бита на пиксель. Можно сжимать с потерями, либо без потерь.

Виды избыточности, которые представлены в изображении.

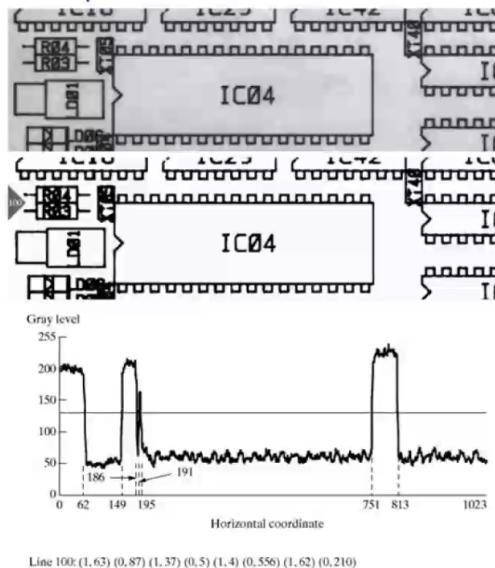
Кодовая избыточность возникает из-за использования кодов, которые не минимизируют среднюю длину кодового слова (равномерный код вместо Хаффмана и т.д.)

Межпиксельная избыточность связана с тем, что пиксель или группы пикселей похожи друг на друга: статистическая зависимость цветовых компонент, локальная схожесть пикселей (local similarity), схожесть удаленных групп пикселей (non-local similarity) (имеются группы пикселей, которые похожи друг на друга (одинаковые буквы)).

Психовизуальная избыточность зрительная система человека имеет разную чувствительность к визуальной информации (разное восприятие вертикальных, горизонтальных и диагональных линий, приоритет одних участков (лиц) изображения над другими).

Машинная избыточность. Если сжимаем для алгоритма, то неважные участки можем сжать сильнее. Например, номера машин оставить четкими.

Наивное сжатие изображения.



Берем монохромное изображение (разрешение 1024 на 343), преобразуем его в бинарное. Для этого значение каждого пикселя заменяем 1, когда оно выше определенного уровня, и 0, когда ниже.

Каждую строку пикселей будем заменять парами (g_i, w_i) , где g_i — значение, а w_i — количество повторений (длина серии).

Каждая серия кодируется 11 битами (10 битов достаточно, чтобы закодировать максимальную длину 1024).

Эффективность сжатия (отношение исходного размера к полученному):

$$C_R = \frac{1024 \cdot 343 \cdot 1}{1266 \cdot 11} = 2.63$$

Объективная оценка качества изображения (пиковое отношение сигнала к шуму (peak signal-to-noise ratio)):

$$PSNR = 10 \lg \frac{|f_{max}(x, y)|^2}{MSE}$$

, где $f_{max}(x, y)$ — максимальное возможное значение яркости, для 8-битного изображения 255.

Среднеквадратичная ошибка:

$$MSE = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f(x, y) - \hat{f}(x, y)]^2$$

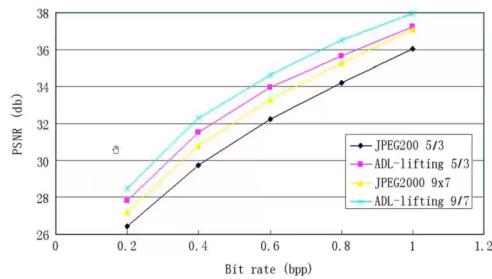
Субъективная:

Визуально сравниваем одно изображение с другим и даем оценку от -3 (немного хуже) до 3 (немного лучше).

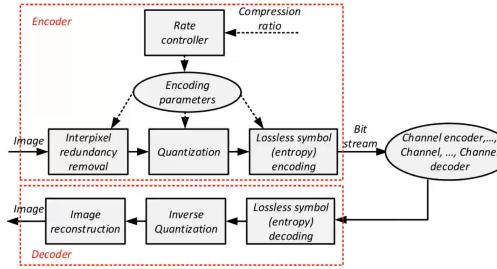
Можно оценивать каждое изображение независимо, по одной или нескольким шкалам.

При субъективной оценке сложно подбирать параметры, их нельзя использовать в кодере.

Сравнение алгоритмов при помощи функции скорость-искажение (в случае с изображением используется bit per pixel):



Общая схема сжатия изображений:



Сначала устранием зависимостей между пикселями, затем выполняется вторичное квантование (первое осуществлялось при переходе сигнала из аналогового в цифровой), затем выполняется энтропийное кодирование (используем коды, которые по своей эффективности приближаются к энтропии источника).

Rate controller выбирает параметры для блоков так, чтобы максимизировать качество.

Наиболее часто используется преобразование из формата RGB24 в формат YCbCr 4:2:0. Прямое преобразование:

$$\begin{cases} Y = 0.299 \cdot R + 0.587G + 0.114 \cdot B, \\ Cb = (B - Y) \cdot 0.5643 + 128, \\ Cr = (R - Y) \cdot 0.7132 + 128. \end{cases}$$

Обратное преобразование:

$$G = Y - 0.714 \cdot (Cr - 128) - 0.334 \cdot (Cb - 128), R = Y + 1.402 \cdot (Cr - 128), B = Y + 1.772 \cdot (Cb - 128).$$

Такое цветовое пространство позволяет уменьшить зависимость между цветовыми компонентами.



Видно, что компоненты яркостная и цветоразностные (которые хранят цветовую информацию) имеют меньше зависимостей. Это позволяет кодировать каждую компоненту независимо друг от друга, как будто это 3 разных изображения. Мы умничили сложность.

Рассмотрим, как сжимать Y. Остальные компоненты сжимаются так же.

Человеческое существо более чувствительно к яркости, чем различия цветов, поэтому цветоразностные компоненты можно уменьшить в 2 раза.

Четыре соседних пикселя имеют общее значение цветоразностных компонент. Теперь пиксель стоит не 24 бит, а 12.

$$\begin{array}{|c|c|} \hline (Y_1, U_1, V_1) & (Y_2, U_2, V_2) \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline (Y_1, U, V) & (Y_2, U, V) \\ \hline \end{array}$$

$$U = \left\lfloor \frac{U_1 + U_2 + U_3 + U_4 + 2}{4} \right\rfloor, V = \left\lfloor \frac{V_1 + V_2 + V_3 + V_4 + 2}{4} \right\rfloor.$$

Будем рассматривать пиксель монохромного изображения как $x_i \in \{0, 1, \dots, 255\}$.

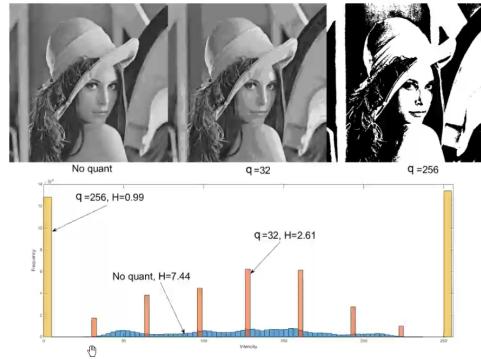
10.1 Простой кодек изображений

На входе монохромное изображение $X = \{x_1, x_2, \dots\}, x_i \in \{0, 1, \dots, 255\}$.

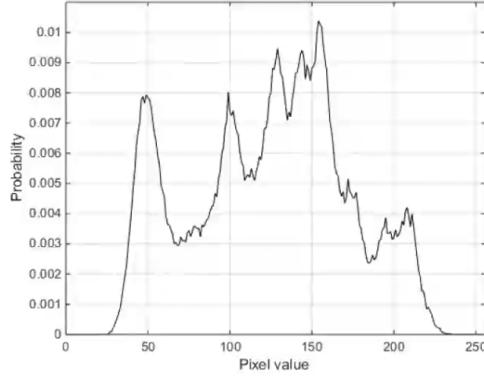
Кодер выполняет квантование $z_i = \lfloor \frac{x_i + \Delta/2}{\Delta} \rfloor$, кодирование $\{z_1, z_2, \dots\}$ (например, кодом Хаффмана). Битрейт R можно оценить как $R \approx H(Z) = -\sum_j p_j \log_2(p_j)$, где p_j это вероятность $z_i = j$.

Декодер выполняет декодирование $\{z_1, z_2, \dots\}$ и восстановление $\hat{x}_i = \Delta \cdot z_i$.

Под квантованием имеется ввиду вторичное квантование, оно позволяет сократить множество значений и улучшить сжатие, но возникают потери:

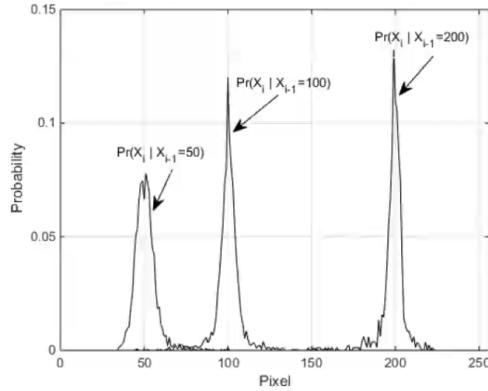


Помним из свойств энтропии, что если вносишься необратимая функция (в данном случае квантование), то энтропия уменьшается. Видно, что при сильном квантовании получаем 1 бит на пиксель (изображение состоит из двух цветов). Если просто кодировать пиксели, например, арифметическим кодом, то получим $H(X) = 7.44$, плюс избыточность кодирования (получается почти 8, как в оригинале). Гистограмма X:



Можно использовать кодирование с предсказанием, чтобы немного улучшить результат.

Можно смоделировать изображение марковским источником с памятью 1 (только предыдущий пиксель). Построим распределения пикселя x_i при заданном пикселе x_{i-1} .



Из максимумов следует, что пиксели изменяются плавно, а если хотим угадывать следующий символ по предыдущему, то лучше всего это получится сделать, если предыдущее значение было 200.

$$H(X_i | 50) = 4.56, H(X_i | 100) = 4.57, H(X_i | 200) = 4.27.$$

Мы можем вычислить $H(X | X)$ и получить все варианты (не только 3): $R(X_i | X_{i-1}) = H(X_i | X_{i-1}) = \sum_{s=0}^{255} p(s)H(X_i | s) = 4.54 < H(X_i) = 7.44$. Получили значение, которое гораздо меньше независимого кодирования. Учитываем предыдущий пиксель в изображении и выигрываем почти 3 бита. РРМ.

Код Хаффмана потребует 256 кодовых таблиц.

10.2 Кодирование с предсказанием

Будем кодировать не сам пиксель, а разность между ним и предыдущим пикслем. Такая разность называется ошибкой предсказания $e_i = x_i - x_{i-1}$:



$$H(E) = 5.07$$

Неправильный алгоритм кодирования. Шаги кодера:

- вычисление ошибки предсказания $e_i = x_i - x_{i-1}$,
- квантование $z_i = \lfloor \frac{e_i + \Delta/2}{\Delta} \rfloor$,
- кодирование z_i .

Шаги декодера:

- декодирование z_i ,
- деквантование $\hat{e}_i = \Delta \cdot z_i$,
- восстановление $\hat{x}_i = \hat{x}_{i-1} + \hat{e}_i$.

Проблема заключается в том, что в декодере накапливается ошибка деквантования. Накопление происходит из-за того, что при восстановлении используется предыдущее значение. В результате восстановления изображения последние пиксели могут иметь сильно завышенные значения (на рисунке последние пиксели становятся белыми):



Чтобы решить эту проблему, можно ограничить накопление ошибки сверху. Для этого разобьем изображение на блоки и будем кодировать каждый независимо.

Блок выберем такой длины, чтобы максимальная ошибка была приемлемой. Нулевой пиксель будем брать среднего значения (128 для 256).

Более эффективный способ решения проблемы заключается в том, чтобы накапливать ошибку и на стороне кодера. Будем вычислять ошибку из оценки предыдущего значения: $e_i = x_i - \hat{x}_{i-1}$, где \hat{x}_{i-1} — это $\hat{x}_i = \hat{x}_{i-1} + \hat{e}_i = \hat{x}_{i-1} + z_i\Delta$, который был восстановлен кодером на предыдущем шаге. Теперь кодер дополнительно выполняет функцию декодера.

Сравнение кодирования с предсказанием и без:



Общая схема кодирования с предсказанием:

В простейшем случае **предсказатель** (предыдущее значение, вычисляется в блоке Prediction) $p_{i+1} = y_i$. В общем случае это линейная функция $p_{i+1} = f(y_0, y_1, \dots, y_i)$.

10.3 JPEG-LS

Обозначим пиксели буквами:

x — текущий пиксель, пиксели до него уже известны, после — еще не прочитаны.

У нас имеется не 1 известный соседний пиксель, а 4. Поэтому, чтобы уменьшить энтропию, при вычислении ошибки будем брать либо пиксель слева, либо пиксель сверху, чтобы ошибка получалась наименьшей. Но декодер не знает, какой пиксель был выбран кодером. Чтобы не передавать информацию о выбранном пикселе в канал, будем использовать пиксель c вместо пикселя x , чтобы решить, считать ошибку по горизонтали или вертикали. Например, если $b - c < a - c$, то кодируем $x - a$.

Человеческое зрение устроено так, что нам легче различать вертикальные и горизонтальные детали, чем диагональные. Поэтому на многих фотографиях яркость будет меняться преимущественно в этих двух направлениях, и описанный подход минимизации ошибки будет работать хорошо.

Используется также пиксель d , чтобы переключать режим работы на более эффективный в данной ситуации (кодирование длин серий), когда яркость не меняется вообще.

Стандарт JPEG-LS работает в 2 режимах: regular mode и run-length mode.

Вычисление градиента:

$$\begin{cases} d_1 = d - b, \\ d_2 = b - c, \\ d_3 = c - a \end{cases}$$

Если $d_1 = d_2 = d_3 = 0$, то кодер переходит в режим кодирования длин серий до тех пор, пока не произойдет $a \neq x$, либо не будет достигнут конец строки. Длина серии передается монотонным кодом (Rice-Golomb code), и кодер возвращается в regular mode.

В regular mode пиксели обрабатываются в растровом порядке. Предсказатель вычисляется по формуле:

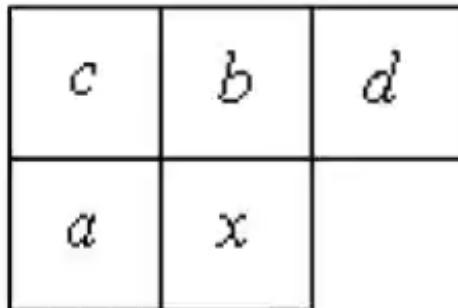
$$p_x \leftarrow \begin{cases} \min(a, b) & if c \geq \max(a, b), \\ \max(a, b) & if c \leq \min(a, b), \\ a + b - c & otherwise. \end{cases}$$

В 3 случае нужно считать среднее арифметическое, но так как операции умножения и деления работают медленно, используем приближенное значение.

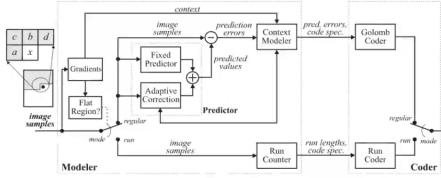
Коррекция предсказателя:

$$p_x \leftarrow p_x + \Delta p(d_1, d_2, d_3).$$

Ошибка предсказания $e_x \leftarrow x - p_x$ кодируется монотонным кодом (Rice-Golomb code).



Общая схема стандарта:



10.4 Кодирование с преобразованием

Делим изображение на непересекающиеся блоки размером $n \times n$.

Выполняем преобразование $n \times n$ для каждого блока.

Квантование устроняет наименее информативные коэффициенты преобразования.

Энтропийное кодирование применяется к квантованным коэффициентам преобразования.

Прямое преобразование:

$$T(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y)g(x, y, u, v),$$

$$u, v = 0, 1, 2, \dots, N - 1.$$

x, y — входное значение по соответствующим координатам. u, v — координаты блока преобразования

Обратное преобразование:

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(x, y)h(x, y, u, v).$$

Сепарельное преобразование:

$$g(x, y, u, v) = g_1(x, u) \cdot g_2(y, v)$$

Симметричное преобразование:

$$g(x, y, u, v) = g_1(x, y) \cdot g_1(u, v)$$

Возможные преобразования:

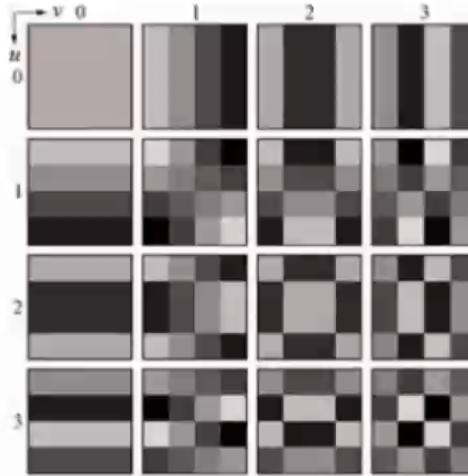
- Карунена-Лоева, гарантирует, что коэффициенты не коррелированы, оптимальный базис зависит от входных данных, его нужно передать декодеру, имеет высокую вычислительную сложность.
- Дискретное преобразование Фурье. Имеет избыточные (мнимые) коэффициенты.

- Дискретное косинусное преобразование.

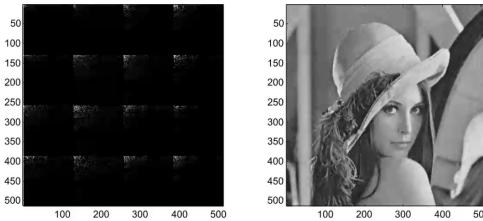
$$g(x, y, u, v) = h(x, y, u, v) = a(u)a(v)\cos\left(\frac{(2x+1)u\pi}{2N}\right)\cos\left(\frac{(2y+1)v\pi}{2N}\right)$$

$$a(u) = \begin{cases} \frac{1}{\sqrt{N}}, & u = 0 \\ \frac{2}{\sqrt{N}}, & u \neq 0. \end{cases}$$

Имели на входе $4 * 4 = 16$ значений, в идеальном случае, когда все значения одинаковые, получили 1 коэффициент преобразования.



После удаления 90% коэффициентов с наименьшей амплитудой:



В блоке наиболее значимые коэффициенты сконцентрированы в углу. Это свойство можно использовать при кодировании. Преобразуем двумерный блок в одномерный вектор так, что в начале будут идти значимые значения, а остальные передавать не будем.

- Уолша-Адамара.

$$g(x, y, u, v) = h(x, y, u, v) = \frac{1}{N}(-1)^{b_i(x)p_i(u)+b_i(y)p_i(v)}$$

$N = 2^m$, $b_i(x)$ — значение бита в x на позиции i .

$$p_0(u) = b_{n-1}(u),$$

$$\begin{aligned}
p_1(u) &= b_{n-1}(u) + b_{n-2}(u), \\
p_2 &= b_{n-2}(u) + b_{n-3}(u), \\
p_{n-1}(u) &= b_1(u) + b_0(u).
\end{aligned}$$

Этапы стандарта *JPEG*:

- Преобразование цветового пространства из *RGB24* в *YCbCr4 : 2 : 0*;
- Разбиение яркостной и цветоразностной компоненты на блоки 8×8 .
- Применение 2-D DCT (двумерное дискретное косинусное преобразование) для каждого блока;
- Квантование DCT коэффициентов;
- DC коэффициент из текущего блока предсказывается при помощи DC коэффициента предыдущего блока и кодирование разности кодом Левенштейна с кодом Хаффмана в первой части кодового слова.
- Сканирование AC коэффициентов в зигзагообразном порядке.
- Одномерный вектор AC кодируется длинами серий и кодом Хаффмана.

Пример. Есть исходный блок 8×8 :

$$X = \begin{pmatrix} 168 & 161 & 161 & 150 & 154 & 168 & 164 & 154 \\ 171 & 154 & 161 & 150 & 157 & 171 & 150 & 164 \\ 171 & 168 & 147 & 164 & 164 & 161 & 143 & 154 \\ 164 & 171 & 154 & 161 & 157 & 157 & 147 & 132 \\ 161 & 161 & 157 & 154 & 143 & 161 & 154 & 132 \\ 164 & 161 & 161 & 154 & 150 & 157 & 154 & 140 \\ 161 & 168 & 157 & 154 & 161 & 140 & 140 & 132 \\ 154 & 161 & 157 & 150 & 140 & 132 & 136 & 128 \end{pmatrix}$$

Вычитаем среднее значение 128 и выполняем 2-D DCT:

$$Y = \begin{pmatrix} 214 & 49 & -3 & 20 & -10 & -1 & 1 & -6 \\ 34 & -25 & 11 & 13 & 5 & -3 & 15 & -6 \\ -6 & -4 & 8 & -9 & 3 & -3 & 5 & 10 \\ 8 & -10 & 4 & 4 & -15 & 10 & 6 & 6 \\ -12 & 5 & -1 & -2 & -15 & 9 & -5 & -1 \\ 5 & 9 & -8 & 3 & 4 & -7 & -14 & 2 \\ 2 & -2 & 3 & -1 & 1 & 3 & -3 & -4 \\ -1 & 1 & 0 & 2 & 3 & -2 & -4 & -2 \end{pmatrix}$$

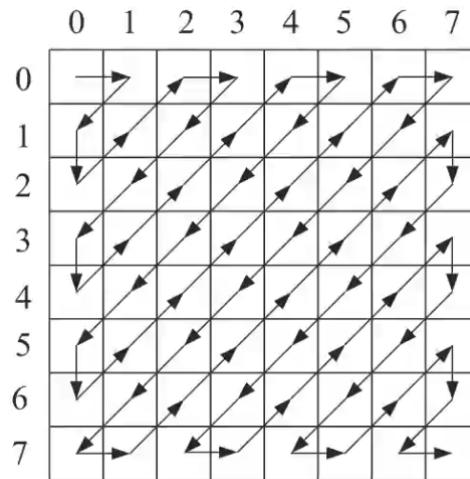
Видно, что коэффициенты с большей амплитудой находятся где-то сверху, слева.

После скалярного квантования:

$$Z = \begin{pmatrix} 13 & 4 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & -2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Остались только самые значимые коэффициенты и много нулей.

Сканируем коэффициенты в зигзагообразном порядке, чтобы значимые коэффициенты оказались в начале вектора:



Вектор после сканирования

$$Z = \{13, 4, 3, 0, -2, 0, 1, 1, 0, 1, -1, -1, 1, 1, 0, \dots, 0\}$$

Первый коэффициент 13 (DC coefficient) предсказывается по DC коэффициенту из предыдущего (слева) блока.

Затем амплитуда ошибки предсказания кодируется монотонным кодом, в котором первая часть кодируется кодом Хаффмана.

Дополнительный бит используется для передачи знака для ненулевых значений.

$$n \rightarrow huff(DCbits)bin(DC)signbit$$

Вычисляем ошибку предсказания для DC, Хаффманом передаем количество бит под DC коэффициент, бинарным кодом передаем вычисленную ошибку без старшего бита (который всегда 1).

Теперь будем кодировать оставшиеся AC коэффициенты (AC coefficients). Представим их парами $(run, level)$, где run — число нулей перед ненулевым коэффициентом, $level$ — значение ненулевого коэффициента:

$(0, 4), (0, 3), (1, -2), (1, 1), (0, 1), (1, 1), (0, -1), (0, -1), (0, 1), (0, 1)$

Кодируем:

$$(run, level) \rightarrow huff(run, levelbitsize)bin(|level|)signbit$$

[run, level bit size] кодируются как один символ уже готовыми таблицами Хаффмана.

Если пара $(run, level)$ не присутствует в кодовой таблице Хаффмана, то передается ESC символ, после чего run и $level$ передаются равномерным кодом.

End-of-block (EOB) символ передается кодом Хаффмана, если в зигзаге далее следуют только нули.