ТЕОРИЯ ИНФОРМАЦИИ

Введение. Задачи, решаемые в рамках теории информации

Теория информации – наука с точно известной датой рождения. Ее появление на свет связывают с публикацией Клодом Шенноном работы “Математическая теория связи” в 1948 г. С тех пор теория информация прошла большой путь, обогатилась огромным числом интересных научных открытий и доказала свою практическую важность. Сегодня в повседневный обиход вошли высокоскоростные модемы для телефонных каналов, лазерные компакт-диски для хранения информации, жесткие диски большой емкости для персональных компьютеров, мобильные телефонные аппараты для сотовых систем связи и многие другие устройства, создание которых было бы невозможно без привлечения методологии и математического аппарата, разработанных в рамках теории информации.

Хотя теории информации часто приписывают несколько более широкое значение, применяя ее методологию в естествознании и искусстве, с точки зрения самого Шеннона, она может корректно рассматриваться только как раздел математической теории связи. Поэтому круг задач теории информации мы поясним с помощью представленной на рис.1.1.1 структурной схемы типичной системы передачи или хранения информации.

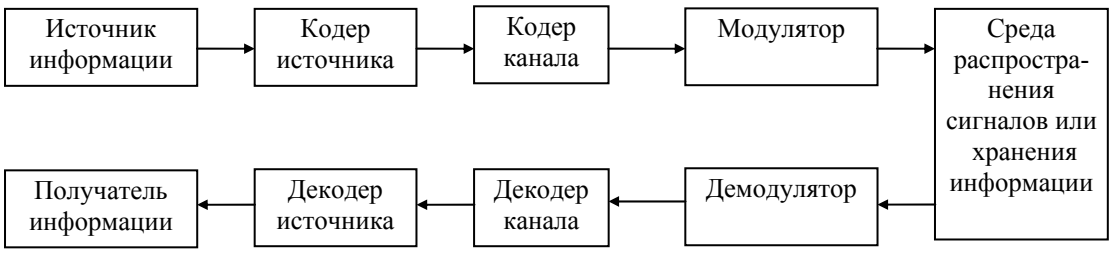


Рис. 1.1.1. Типичная система передачи (хранения) информации

В этой схеме под источником понимается любое устройство или объект живой природы, порождающие сообщения, которые должны быть перемещены в пространстве или во времени. Это может быть клавиатура компьютера, человек, аналоговый выход видеокамеры и т. п. Поскольку, независимо от изначальной физической природы, все подлежащие передаче сообщения обычно преобразуется в форму электрических сигналов, именно такие сигналы мы и будем рассматривать как выход источника.

Цель кодера источника – представление информации в наиболее компактной форме. Это нужно для того, чтобы эффективно использовать ресурсы канала связи либо запоминающего устройства.

Далее следует кодер канала, задачей которого является обработка информации с целью защиты сообщений от помех при передаче по каналу связи либо возможных искажений при их хранении. Модулятор служит для преобразования сообщений, формируемых кодером канала, в сигналы, согласованные с физической природой канала связи или средой накопителя информации.

Остальные блоки, расположенные на приемной стороне, выполняют обратные операции и предоставляют получателю информацию в удобном для использования виде.

Итак, в теории информации можно выделить следующие разделы:

1. *Кодирование дискретных источников.* Иногда эту часть теории информации называют кодированием без потерь, кодированием для канала без шума, сжатием информации.

2. *Кодирование информации для передачи по каналу с шумом.* Речь идет о защите информации от помех в каналах связи.

3. *Кодирование с заданным критерием качества.* В некоторых системах связи искажения информации допустимы. Более того, информация аналоговых источников вообще не может быть представлена в цифровой форме без искажений. В данном разделе речь идет о методах кодирования, обеспечивающих наилучший компромисс между качеством (оцениваемым некоторой объективной мерой искажения) и затратами на передачу информации. Сегодня задачи такого типа стали особенно актуальны, поскольку они находят широкое применение для цифровой передачи речи, видео- и аудиоинформации.

4. *Кодирование информации для систем со многими пользователями.* Здесь обсуждается оптимальное взаимодействие абонентов, использующих какой-либо общий ресурс, например, канал связи Примеры систем с множественным доступом - системы мобильной связи, локальные сети ЭВМ.

5. *Секретная связь, системы защиты информации от несанкционированного доступа*. Здесь также можно указать широкий круг актуальных задач, лежащих на стыке теории информации, теории вычислительной сложности алгоритмов, исследования операций, теории чисел.

Из перечисленных разделов в данном пособии кратко рассматриваются первые три.

Раздел 1.

Энтропия дискретных источников.

1.1. Дискретные источники сообщений

Цель этого параграфа – напомнить те понятия, определения, формулы теории вероятностей, которыми мы будем пользоваться в дальнейшем.

Будем обозначать через X некоторое дискретное множество, т.е. множество, содержащее конечное или счетное число элементов $x \in X$. Множество X назовем *множеством событий*, его элементы – *элементарными событиями.* Непустые подмножества $A \subset X$ мы называем событиями, через $|A|$ мы обозначаем мощность множества A (число элементов в A) . Если $|A| > 1$, *событие называется сложным событием.* В рамках теории информации в качестве X рассматривается множество сообщений, его подмножества являются сообщениями, элементарные события – элементарными сообщениями.

Напомним, что произвольное множество чисел $\{p(x)\}, x \in X$ задает *распределение вероятностей* на дискретном множестве $X = \{x\}$, если все числа неотрицательны и удовлетворяют условию нормировки $\sum_{x \in X} p(x) = 1$.

Вероятность сложного события A определяется как

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

Множество сообщений X в совокупности с распределением вероятностей $\{p(x)\}, x \in X$ образуют *дискретный вероятностный ансамбль* $X = \{x, p(x)\}$.

Рассмотрим множество Ω всевозможных подмножеств множества X. Заметим, что если X - конечное множество, содержащее $|X| = N$ элементов, то различных подмножеств будет $|\Omega| = 2^N$. Элементами Ω являются элементарные и сложные события из X, пустое (невозможное) событие Ø, достоверное событие X. Для любого $A \in \Omega$ определена операция взятия дополнения A^c, для пар множеств $A, B \in \Omega$ определены операции объединения событий $A \bigcup B$ и пересечения событий $A \bigcap B$. Иначе их называют соответственно суммой и произведением событий. Для произведения событий мы будем

пользоваться обозначением $AB = A \bigcap B$. Перечисленные операции над событиями определяют множество Ω как *булеву алгебру* событий.

Справедливы формулы

$$P(\O) = 0;\ P(X) = 1;$$
$$P(A^c) = 1 - P(A);$$
$$P(A \bigcup B) = P(A) + P(B) - P(AB).$$

События *несовместны*, если их пересечение – пустое множество. Для несовместных событий вероятность объединения равна сумме вероятностей событий. Если условие несовместности не выполняется, вероятность объединения становится меньше суммы вероятностей . Это верно для любого числа событий. Оценка

$$P(\bigcup_{m=1}^{M} A_m) \le \sum_{m=1}^{M} P(A_m)$$

называется *аддитивной оценкой* вероятности суммы событий. Равенство в этой оценке имеет место только для несовместных событий.

Для произвольной пары событий $A, B \subseteq X$ *условной вероятностью* события A при условии B называется величина

$$P(A \mid B) = \frac{P(AB)}{P(B)}$$

при $P(B) \neq 0$ и величина $P(A \mid B) = 0$ при $P(B) = 0$.

Из этого определения следует

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

Для совокупности из произвольного числа событий из этой формулы можно получить соотношение

$$P(A_1 ... A_n) = P(A_1)P(A_2 \mid A_1)P(A_3 \mid A_1 A_2)...P(A_n \mid A_1 ... A_{n-1}).$$

События $A, B \subseteq X$ называют *независимыми,* если

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

Совместно независимыми называют события $A_1, ..., A_n \subseteq X$ такие, что

$$P(A_1 ... A_n) = P(A_1)P(A_2)...P(A_n).$$

Если $A, B \subseteq X$ - независимые события, то

$$P(A \mid B) = P(A);\quad P(B \mid A) = P(B).$$

Пусть несовместные события $H_1, ..., H_M$ таковы, что $P(\bigcup_{m=1}^{M} H_m) = 1$. Такой набор событий в иногда удобно рассматривать как систему *гипотез* по отношению к заданному событию A. Имеет место *формула полной вероятности*

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

и *формула апостериорной вероятности или формула Байеса*

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

Произведением множеств X и Y называется множество $Z = XY = \{(x, y) : x \in X, y \in Y\}$, то есть множество упорядоченных пар, первый элемент которых принадлежит множеству X, второй - множеству Y. Хотя обозначения для произведения множеств и произведения событий совпадают, смысл этих понятий, очевидно, различен. Какое из двух понятий имеется в виду всегда ясно из контекста.

Нетрудно подсчитать число элементов в множестве $Z = XY$:

$$| XY | = | X | \cdot | Y |.$$

Перемножая произвольное число n множеств X_1, X_2,…, X_n, получим множество последовательностей длины n, составленных из элементов множеств-сомножителей. В частном случае, когда все множества одинаковы, $X_1 = X_2 = ... = X_n = X$, используется сокращенная запись

$$X_1 X_2 .. X_n = X^n = \{(x_1,...,x_n) : x_i \in X, i = 1,...,n\}.$$

Чтобы определить *произведение вероятностных ансамблей* $X = \{x, p_X(x)\}$ и $Y = \{y, p_Y(y)\}$, нужно на произведении множеств XY задать совместное распределение вероятностей $\{p_{XY}(x,y)\}$. Получим ансамбль $XY = \{(x,y), p_{XY}(x,y)\}$. Нижний индекс при распределениях вероятностей всякий раз указывает на то, на каком множестве определено данное распределение. Мы будем опускать этот индекс всякий раз, когда это не приводит к неоднозначному пониманию.

Это определение естественным образом распространяется на произвольное число ансамблей-сомножителей. Число сомножителей n называют размерностью ансамбля-произведения, соответствующее распределение вероятностей – многомерным (n-мерным). Если задано n-мерное распределение, тем самым заданы и распределения для меньших размерностей. В частности, для произведения двух ансамблей имеем

$$p(x) = \sum_{y \in Y} p(x,y),$$
$$p(y) = \sum_{x \in X} p(x,y).$$

Условным распределением вероятностей на элементах множества X при фиксированном элементе $y \in Y$ называется распределение

$$p(x \mid y) = \begin{cases} \dfrac{p(x,y)}{p(y)}, & \text{если } p(y) \neq 0, \\ 0 & \text{в противном случае;} \end{cases} \qquad x \in X,$$

Ансамбли X и Y независимы, если имеют место тождества

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

Используя эти понятия, легко обобщить формулу перемножения вероятностей, формулу полной вероятности и формулу Байеса на произведение вероятностных ансамблей. Например, справедлива формула

$$p(x_1,...,x_n) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1 x_2) \cdot ... \cdot p(x_n \mid x_1,...,x_{n-1}).$$

Большую роль в теории информации играют случайные ансамбли, элементы которых – числа. Такие ансамбли являются *случайными величинами.*

Математическим ожиданием $x \in X$ называется

$$\mathbf{M}_X[x] = \sum_{x \in X} x p(x).$$

Для произвольного множества X можно определить функцию $\varphi(x)$, $x \in X$, принимающую вещественные значения. Тем самым устанавливается отображение X на множество вещественных чисел. Новый вероятностный ансамбль с множеством значений $Y = \{y = \varphi(x)\}$ является случайной величиной. Для вычисления математического ожидания величины y необязательно знать распределение вероятностей $p_Y(y)$ для y. Зная распределение $p_X(x)$ на X, величину $\mathbf{M}_Y[y]$ можно получить, воспользоваться следующим полезным свойством математического ожидания

$$\mathbf{M}_Y[y] = \mathbf{M}_X[\varphi(x)] = \sum_{x \in X} \varphi(x) p_X(x). \qquad (1.1.1)$$

Действительно,

$$\mathbf{M}_Y[y] = \sum_{y \in Y} y p_Y(y) = \sum_{y \in Y} y \sum_{x : \varphi(x) = y} p_X(x) = \sum_{x \in X} \varphi(x) p_X(x).$$

Другими важными характеристиками случайных величин являются *дисперсия*

$$\mathbf{D}_X[x] = \mathbf{M}_X\left[(x - \mathbf{M}_X[x])^2\right]$$

и *корреляционный момент*

$$K_{XY}(x, y) = \mathbf{M}_{XY}\left[(x - \mathbf{M}[x])(y - \mathbf{M}[y])\right].$$

В определениях математического ожидания, дисперсии и корреляционного момента нижние индексы указывают на ансамбли, по которым производится усреднение. Для сокращения записи мы не будем писать этот индекс, если из контекста понятно, о каком ансамбле идет речь.

Если для двух случайных величин корреляционный момент равен нулю, случайные величины называют некоррелированными. Напомним следующие свойства числовых характеристик

1. Для любых случайных величин x и y
$$\mathbf{M}[x + y] = \mathbf{M}[x] + \mathbf{M}[y].$$
2. Если c - константа, x - случайная величина, то
$$\mathbf{M}[cx] = c\mathbf{M}[x].$$
3. Если x и y - независимые случайные величины, то
$$\mathbf{M}[xy] = \mathbf{M}[x]\mathbf{M}[y].$$
4. Для некоррелированных случайных величин x и y
$$\mathbf{D}[x + y] = \mathbf{D}[x] + \mathbf{D}[y].$$
5. Если c - константа, x - случайная величина, то
$$\mathbf{D}[cx] = c^2\mathbf{D}[x].$$
6. Если c - константа, x - случайная величина, то
$$\mathbf{D}[c + x] = \mathbf{D}[x].$$
7. Если x и y независимы то $K(x, y) = 0$, то есть из независимости случайных величин следует их некоррелированность. Обратное, вообще говоря, неверно.

Доказательство этих свойств – простое и полезное упражнение на применение правила (1.1.1).

1.2. Измерение информации. Собственная информация.

Очевидно, задача количественного измерения информации возникла при решении конкретных практических задач. Можно предположить, что первоначальным мотивом было стремление уменьшить количество электрических сигналов, необходимых для передачи сообщений по каналам связи. Поэтому разумной мерой информации, содержащейся в сообщении является мера, монотонно связанная с затратами на передачу сообщения.

Условимся о том, что, по крайней мере с точки зрения получателя информации, сообщения представляют собой случайные события. Рассмотрим в качестве источника произвольное дискретное множество X и каждой букве $x \in X$ припишем вероятность $p(x)$.

Сформулируем интуитивные требования к некоторой мере $\mu(x)$, определенной для всех $x \in X$, которую следует принять как меру информации, содержащейся в сообщениях ансамбля $X = \{x, p(x)\}$.

- Поскольку предполагается, что эта мера будет определять затраты, связанные с передачей или хранением сообщений, мера должна быть неотрицательной.
- Поскольку для нас несущественно, каким образом будут интерпретированы и использованы передаваемые сообщения, мера должна однозначно определяться только

вероятностью сообщения. Поэтому вместо $\mu(x)$ будем писать $\mu(p(x))$.

• Установим характер зависимости меры от вероятности сообщений. Пусть имеется множество из двух равновероятных сообщений и стоит задача кодирования сообщений этого множества двоичными символами алфавита $A=\{0,1\}$. Интуитивно ясно, что неплохой способ кодирования состоит в том, чтобы сопоставить одному из сообщений символ 0, другому - символ 1. Точно также для кодирования 4 равновероятных сообщений можно использовать последовательности длины 2, для 8 сообщений - длины 3 и т.д. В этих примерах вероятностям 1/2, 1/4, 1/8,... соответствовали затраты на передачу равные соответственно 1, 2, 3,... единицам информационной меры. Следовательно, разумно потребовать, чтобы мера информации удовлетворяла соотношению $\mu(p^m)=m\mu(p)$.

• Рассмотрим последовательность сообщений $x_1,...,x_n$, выбираемых независимо из одного и того же множества X с одним и тем же распределением вероятностей $\{p(x)\}$. Из статистической независимости сообщений следует, что знание предыдущих сообщений не помогает предугадать, какими будут следующие. Поэтому затраты на передачу последовательности складываются из затрат на передачу каждого отдельного сообщения. Получаем еще одно разумное требование к информационной мере: $\mu(x_1,...,x_n)=\mu(x_1)+...+\mu(x_n)$ для независимых сообщений $x_1,...,x_n$.

Перечисленные требования к информационной мере приводят нас к следующему определению:

Собственной информацией $I(x)$ сообщения x, выбираемого из дискретного ансамбля $X=\{x,p(x)\}$, называется величина, вычисляемая по формуле

$$I(x)=-\log p(x). \qquad (1.2.1)$$

В этой формуле не указано основание логарифма. Мы и дальше не будем его указывать, всякий раз подразумевая, что логарифмы вычисляются по основанию 2, если не оговорено другое. Это соответствует измерению информации в **битах**. Если бы логарифм вычислялся по натуральному основанию, единицей измерения информацией стал бы **нат.** Если основание десятичное - информация измеряется в **хартли** (в честь Хартли, который использовал формулу (1.2.1) еще до Шеннона). Биты удобнее, поскольку сразу показывают, сколько двоичных символов надо потратить на передачу сообщения. В теоретико-информационных исследованиях часто предпочитают наты, поскольку $\ln x$ удобнее дифференцировать, чем $\log x$. Мы будем пользоваться битами.

Из определения собственной информации и свойств логарифма непосредственно вытекают следующие свойства собственной информации.

1. Неотрицательность: $I(x)\ge 0, x\in X$.
2. Монотонность: если $x_1,x_2\in X$, $p(x_1)\ge p(x_2)$, то $I(x_1)\le I(x_2)$
3. Аддитивность. Для независимых сообщений $x_1,...,x_n$ имеет место равенство

$$I(x_1,...,x_n)=\sum_{i=1}^{n} I(x_i).$$

Примеры.

1. Пусть $X=\{a,b,c,d\}$ и вероятности букв равны $p(a)=1/2$, $p(b)=1/4$, $p(c)=p(d)=1/8$. Тогда собственные информацией букв равны $I(a)=1$, $I(b)=2$, $I(c)=I(d)=3$ бит. Легко указать способ кодирования, при котором на каждую букву будет затрачено по два бита. Наша мера информации показывает, что на самом деле буквы несут различное количество информации, и, может быть, разумнее тратить различное число бит на передачу различных букв (как в азбуке Морзе). Позже мы вернемся к этому примеру и проверим наше предположение.

2. Пусть $X=\{a,b\}$ и вероятности букв равны $p(a)=0.05$, $p(b)=0.95$. Тогда $I(a)=4{,}322$, $I(b)=0{,}216$. Мы получили дробные числа. Более того, передача

одного из двух символов может потребовать больше 4 бит информации! На второй символ предлагается тратить примерно 1/5 бита. Это, на первый взгляд, кажется невозможным. Однако, как мы увидим позже, тратить целый бит на передачу каждой буквы рассматриваемого источника - непозволительная роскошь.

Нельзя ли было использовать для измерения информации другую функцию? Этот вопрос поднимался как на заре теории информации, так и в последние годы. Доказано, что при предположениях, сформулированных выше и некоторых других предположениях, касающихся непрерывности и дифференцируемости информационной меры, информационная мера единственна с точностью до постоянного множителя, т.е. с точностью до выбора основания логарифма.

1.3. Энтропия

Определенная в предыдущем параграфе мера информации, содержащейся в сообщении, представляет собой случайную величину. Собственная информация сообщения x дискретного ансамбля $X = \{x, p(x)\}$ характеризует "информативность" или "степень неожиданности" конкретного сообщения. Естественно, среднее значение или математическое ожидание этой величины по ансамблю $X = \{x, p(x)\}$ будет характеристикой информативности всего ансамбля.

Энтропией дискретного ансамбля $X = \{x, p(x)\}$ называется величина

$$H(X) = \mathbf{M}\big[-\log p(x)\big] = -\sum_{x \in X} p(x) \log p(x) .$$

Можно интерпретировать энтропию как количественную меру априорной неосведомленности о том, какое из сообщений будет порождено источником. Часто говорят, что энтропия является мерой неопределенности. Приведем несколько свойств энтропии. Эти свойства дополнительно проясняют смысл этого понятия и позволяют оценивать энтропию ансамбля, не выполняя точных вычислений.

Свойство 1.3.1. $H(X) \geq 0$.

Свойство 1.3.2. $H(X) \leq \log|X|$. Равенство имеет место в том и только в том случае, когда элементы ансамбля X равновероятны.

Свойство 1.3.3. Если для двух ансамблей X и Y распределения вероятностей представляют собой одинаковые наборы чисел (отличаются только порядком следования элементов), то $H(X) = H(Y)$.

Свойство 1.3.4. Если ансамбли X и Y независимы, то $H(XY) = H(X) + H(Y)$.

Свойство 1.3.5. Энтропия – выпуклая $\cap$ функция распределения вероятностей на элементах ансамбля X.

Свойство 1.3.6. Пусть $X = \{x, p(x)\}$ и $A \subseteq X$. Введем ансамбль $X' = \{x, p'(x)\}$, задав распределение вероятностей $p'(x)$ следующим образом

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

Тогда $H(X') \geq H(X)$. Иными словами, «выравнивание» вероятностей элементов ансамбля приводит к увеличению энтропии.

Свойство 1.3.7. Пусть задан ансамбль X и на множестве его элементов определена функция $g(x)$. Введем ансамбль $Y = \{y = g(x)\}$. Тогда $H(Y) \leq H(X)$. Равенство имеет место тогда и только тогда, когда функция $g(x)$. обратима.

Смысл последнего утверждения состоит в том, что обработка информации не приводит к увеличению энтропии.

Доказательства свойств 1.3.1, 1.3.3 и 1.3.4 мы опускаем как очень простое упражнение для читателя. Доказательства свойств 1.3.5 и 1.3.6 будут даны в следующем параграфе. Доказательство свойства 1.3.7 отложим до знакомства с понятием условной энтропии (п.1.5).

Доказательство свойства 1.3.2. Рассмотрим разность левой и правой частей доказываемого неравенства:

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

$$= \sum_{x \in X} p(x) \log \frac{1}{p(x)|X|} \le \log e \left[\sum_{x \in X} p(x) \left(\frac{1}{p(x)|X|} - 1 \right) \right] =$$

$$= \log e \left(\sum_{x \in X} \frac{1}{|X|} - \sum_{x \in X} p(x) \right) = 0.$$

Поясним приведенные выкладки. Первый переход основан на свойстве нормировки вероятностей, второй – на свойствах логарифма. Далее использовано хорошо известное неравенство

$$\ln x \le x - 1,$$

эквивалентное неравенству

$$\log x \le (x-1) \log e. \qquad (1.3.1)$$

Нам придется еще не раз воспользоваться этим свойством логарифма. Поэтому остановимся на нем подробнее. Графики функций $\ln x$ и $(x-1)$ приведены на рис.1.3.1. Из рисунка видно, что неравенство превращается в равенство только в одной точке $x = 1$. Поэтому в нашем случае необходимым и достаточным условием равенства будет условие $p(x) = 1/|X|$, $x \in X$. Последующие переходы снова используют условие нормировки вероятностей. Итак, разность оказалась неотрицательной и, следовательно, свойство 1.3.2 доказано. ■

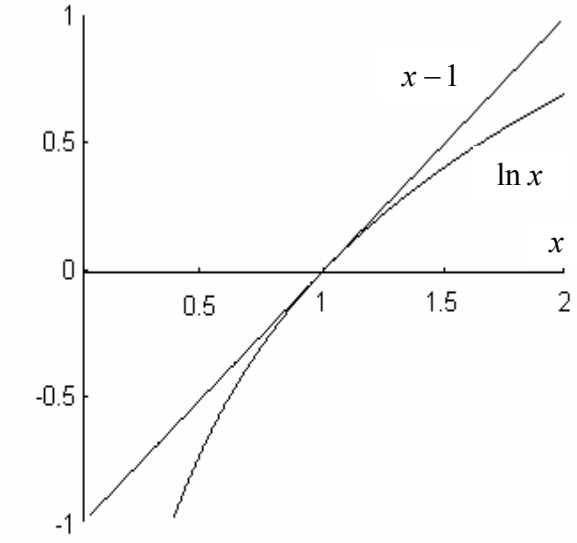


Рис. 1.3.1. Графическая интерпретация неравенства $\ln x \le x - 1$

Сформулированные выше свойства имеют простую интерпретацию, если вспомнить, что энтропия определяет информативность источника или средние затраты на представление информации в двоичной форме. Рассмотрим, например, источник с объемом алфавита равным 256. Понятно, что при любом распределении вероятностей на буквах источника одного байта (8 бит) на букву достаточно для хранения сообщений источника. Свойство 1.3.2 говорит о том, что в предельном случае, когда все буквы источника равновероятны, энтропия источника как раз и равна 8 бит. Предположим теперь, что речь идет о текстовом файле на русском языке. Если использовать в качестве вероятностей букв их частоты появления в типичном тексте, то, подставив вероятности в формулу для энтропии, получим величину близкую к 4,5. Это означает, что буква русского текста несет в среднем 4,5 бита и не следует тратить для ее хранения целый байт. Попробуем обработать файл каким-либо стандартным архиватором. Окажется, что файл достаточно

большого объема сожмется примерно в 4 раза. Получается, что на самом деле и энтропия – плохая оценка затрат на хранение букв источника. Это не совсем верно. Обратимся к свойству 1.3.4. Из него следует: что затраты на хранение последовательности букв можно подсчитать как сумму затрат на каждую букву в том случае, когда буквы независимы. Это условие заведомо не выполняется для осмысленного текста. Значит, нам необходимо научиться подсчитывать энтропию источников зависимых сообщений. Этим мы займемся позже. Свойство 1.3.6 подсказывает, почему для одних ансамблей энтропия велика, для других – мала. Все дело в том, что, буквы имеют разные вероятности. Зная вероятности, можно прогнозировать появление букв. Это все равно, что мы что-то уже имеем какую-то информацию еще до того, как буква порождена источником. Нужно только научиться использовать на практике эту априорную информацию.

Рассмотрим теперь очень простой и очень важный пример – двоичный ансамбль сообщений.

Пример. Рассмотрим двоичный ансамбль $X = \{0,1\}$. Положим $p(1) = p$, $p(0) = 1 - p = q$. Энтропия этого ансамбля равна

$$H(X) = -p\log p - q\log q = h(p).$$

Мы ввели специальное обозначение $h(p)$ для энтропии двоичного ансамбля, в котором один из элементов имеет вероятность p. Эта функция будет часто использоваться. Построим график зависимости $h(p)$ от p при $p \in [0,1]$. Начнем с крайних точек $p = 0$ и $p = 1$. В обоих случаях имеет место неопределенность. Тем не менее, используя правило Лопиталя, легко вычислить предельные значения $h(p)$ при $p \to 0$ и $p \to 1$. Получим $h(0) = h(1) = 0$. Далее, заметим, что $h(p) = h(1-p)$, то есть функция симметрична относительно оси $p = 1/2$. Это можно было бы также утверждать, опираясь на свойство 3. Чтобы найти экстремальные точки, следует вычислить производную функции $h(p)$. Производная равна

$$h'(p) = -\log p + \log(1-p).$$

Точкой экстремума оказывается точка $p = 1/2$. Вычисляя вторую производную

$$h''(p) = -\log e / p - \log e/(1-p) < 0,$$

убеждаемся в том, что это точка максимума. Этот результат можно было предсказать на основе свойства 1.3.2. Кроме того, нетрудно видеть, что вторая производная строго отрицательна. Отсюда следует, что энтропия двоичного ансамбля – строго выпуклая вверх функция параметра p. График функции $h(p)$ представлен на рис.1.3.2. ■

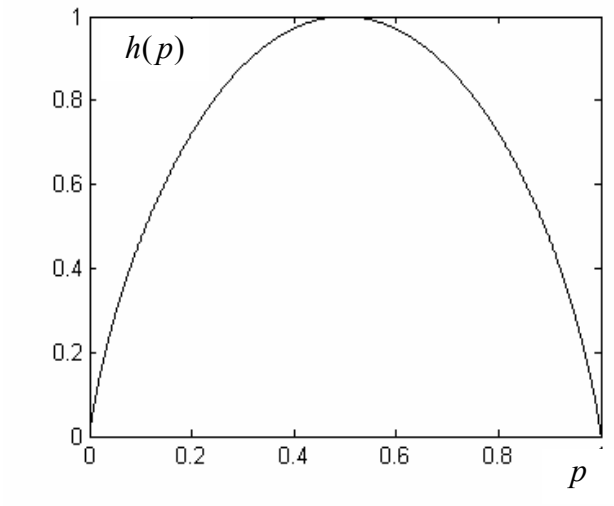


Рис. 1.3.2. График функции $h(p)$

Программисты и инженеры, имеющие дело с информацией, представленной в двоичной форме, привыкли называть битом любой двоичный разряд в записи числа, содержимое двоичной ячейки памяти или значение двоичного сигнала. Мы теперь знаем, что в терминах теории информации один бит – максимальное среднее количество информации, переносимое двоичным сигналом. Если, например, вероятность единицы равна 0,1, то энтропия двоичного ансамбля равна 0,469, то есть приблизительно половине бита. Получается, что «инженерный бит» в

данном случае в 2 раза больше «теоретико-информационного бита».

1.4. Выпуклые функции многих переменных

Нам предстоит исследовать поведение некоторых функций на множестве X в зависимости от распределения вероятностей на X. Это означает, что мы будем иметь дело с функциями нескольких переменных. Поведение функции одной переменной мы привыкли изучать с помощью производных. По знаку первой производной мы определяем, возрастает функция в данной точке или убывает. Равенство нулю первой производной означает, что точка подозрительна на экстремум. Этот подход заведомо не может быть использован для анализа функций многих переменных.

Нам поможет понятие выпуклых функций. В математической литературе на русском языке различают выпуклые и вогнутые функции. Даже среди тех, кто уверен в том, что точно знает, какие функции выпуклые, какие – вогнутые, нет абсолютного единства мнений по этому вопросу. Поэтому, в теории информации принято различать функции выпуклые вверх и функции выпуклые вниз. В первом случае для краткости пишут, что функция выпукла $\cap$, а во втором что она выпукла $\cup$.

В следующих ниже определениях мы опираемся, в основном, на геометрические представления о выпуклости, а не на вычисление производных. Прежде всего, рассмотрим множество значений аргументов (область определения функции) и введем для него понятие выпуклости.

Множество аргументов функции запишем в виде вектора. Рассмотрим произвольную функцию $f(\mathbf{x})$, определенную для всех векторов из некоторого множества $R = \{\mathbf{x}\}$. Само множество R мы считаем подмножеством m-мерного евклидова пространства векторов с вещественными компонентами. На рис. 1.4.1 приведены примеры двумерных областей R.

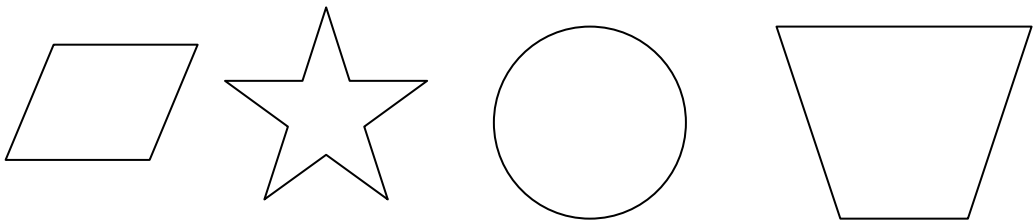


Рис.1.4.1. Примеры двумерных областей определения функций

Среди этих примеров множеств второе множество невыпуклое, остальные 3 – выпуклые. Формальный критерий – следующий. Если для любых двух точек множества весь отрезок, соединяющий эти точки, принадлежит множеству, множество выпукло. Формальное определение следующее.

Множество вещественных векторов R *выпукло*, если для любых $\mathbf{x}, \mathbf{x}' \in R$ и любого $\alpha \in [0,1]$ вектор $\mathbf{y} = \alpha\mathbf{x} + (1-\alpha)\mathbf{x}'$ принадлежит R.

В этом определении вектор $\mathbf{y}$ представляет собой одномерную линейную функцию параметра α или, проще говоря, прямую. Эта прямая, очевидно, проходит через точки $\mathbf{x}, \mathbf{x}'$. Поскольку область изменения параметра α ограничена отрезком $[0,1]$, множество значений вектора $\mathbf{y}$ принадлежит отрезку, соединяющему точки $\mathbf{x}$ и $\mathbf{x}'$. Таким образом, формальное определение в точности соответствует графической интерпретации.

Приведем важный для нас содержательный пример выпуклой области.

Утверждение 1.4.1. Множество вероятностных векторов длины M выпукло.

Доказательство. Напомним, что компоненты вероятностных векторов неотрицательны и сумма компонент равна 1. Для двух распределений вероятностей $\mathbf{p} = (p_1,..., p_M)$ и $\mathbf{p}' = (p_1',..., p_M')$ и параметра $\alpha \in [0,1]$ рассмотрим множество векторов вида

$$\mathbf{q} = \alpha\mathbf{p} + (1-\alpha)\mathbf{p}' .$$

Все элементы вектора $\mathbf{q} = (q_1,..., q_M)$ неотрицательны, поскольку в правой части имеем сумму двух неотрицательных векторов. Сумма компонент вектора $\mathbf{q}$ равна

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

Тем самым утверждение доказано.■

Теперь мы готовы к тому, чтобы конкретизировать понятие выпуклой функции.

Функция $f(\mathbf{x})$ векторного аргумента $\mathbf{x}$, определенная на выпуклой области R, называется *выпуклой* $\cap$ на этой области, если для любых $\mathbf{x}, \mathbf{x}' \in R$ и любого $\alpha \in [0,1]$ имеет место неравенство

$$f(\alpha\mathbf{x} + (1-\alpha)\mathbf{x}') \geq \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{x}') \ . \qquad (1.4.1)$$

Отметим, что если в (1.4.1) возможно равенство при некоторых значениях $\alpha \in (0,1)$, функцию называют *нестрого выпуклой*. В противном случае она называется *строго выпуклой.* Выпуклость области определения функции нужна для того, чтобы выражение в левой части (1.4.1) имело смысл.

Понятно, что определение функции выпуклой $\cup$ получается из (1.3) изменением знака неравенства на противоположный. Мы в дальнейшем рассматриваем только выпуклые $\cap$ функции, считая, что перенос результатов на второй случай не составляет труда.

Рис.1.4.2 поясняет понятие выпуклой функции многих переменных на примере функции одной переменной. Из элементарных геометрических соображений следует, что правой части (1.4.1) при различных значениях α соответствуют точки отрезка, соединяющего точки с координатами $(x_1, f(x_1))$ и $(x_2, f(x_2))$. Левой части соответствуют точки на кривой. Из рисунка видно, что определение функции многих переменных хорошо согласуется с привычным понятием выпуклости функции одной переменной.

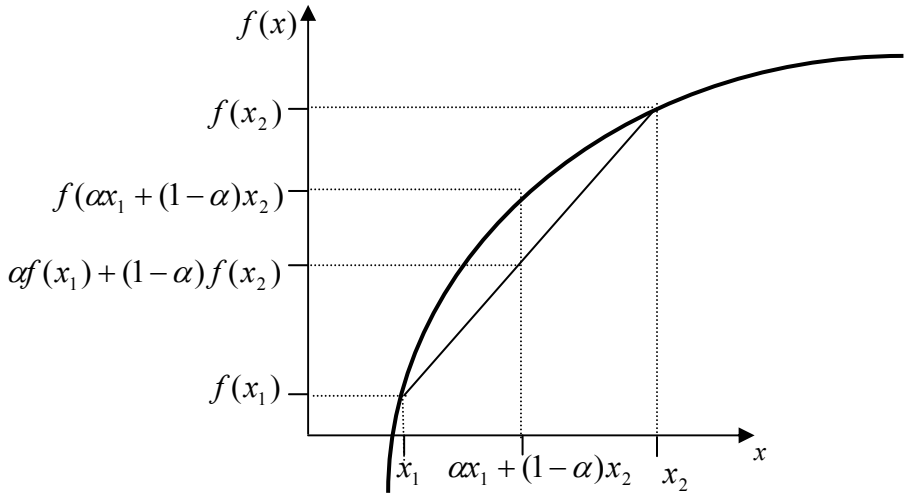


Рис. 1.4.2. Пример выпуклой функции

Непосредственно из определения выпуклой функции, используя метод математической индукции, легко вывести следующее свойство выпуклых функций

Свойство 1.4.1. Пусть $f(\mathbf{x})$ – выпуклая $\cap$ функция векторного аргумента $\mathbf{x}$, определенная на выпуклой области R и пусть константы $\alpha_1,...,\alpha_M \in [0,1]$ таковы, что $\sum_{m=1}^{M}\alpha_m = 1$. Тогда для любых $\mathbf{x}_1,...,\mathbf{x}_M \in R$

$$f\left(\sum_{m=1}^{M}\alpha_m\mathbf{x}_m\right) \geq \sum_{m=1}^{M}\alpha_m f(\mathbf{x}_m). \qquad (1.4.2)$$

Это свойство имеет следующую важную для дальнейшего применения интерпретацию. Константы α_m, использованные в формулировке этого свойства, можно интерпретировать как вероятности соответствующих векторов $\mathbf{x}_m$. Тогда суммы в правой и левой части (1.4.2) можно интерпретировать как математические ожидания, если считать, что случайный вектор $\mathbf{x}$ принимает значение $\mathbf{x}_m$ с вероятностью α_m. Получаем полезное неравенство

$$f\big(\mathbf{M}[\mathbf{x}]\big) \geq \mathbf{M}\big[f(\mathbf{x})\big]. \qquad (1.4.3)$$

Именно это простое неравенство в дальнейшем сократит выкладки, связанные с анализом информационных характеристик алгоритмов кодирования. Однако, чтобы пользоваться им, нужно быть уверенным, что функция выпукла. Поэтому остаток параграфа посвятим выводу признаков выпуклости функций многих переменных. В формулировках следующих утверждений, вытекающих из определения выпуклых функций, предполагается, что все функции определены на одной и той же выпуклой области.

Свойство 1.4.2. Сумма выпуклых функций выпукла.

Свойство 1.4.3. Произведение выпуклой функции и положительной константы представляет собой выпуклую функцию.

Прямым следствием этих утверждений является еще одно свойство.

Свойство 1.4.4. Линейная комбинация выпуклых функций с неотрицательными коэффициентами – выпуклая функция.

Теперь мы знаем, что доказать выпуклость функции многих переменных можно, представив ее в виде линейной комбинации функций, выпуклость которых доказать легко. Если, например, функция представлена в виде линейной комбинации функций одной переменной, то анализ составляющих можно выполнить с помощью вычисления производных. Опираясь на эти результаты, докажем выпуклость энтропии ансамбля как функции распределения вероятностей на множестве его элементов.

Теорема 1.4.1. Энтропия $H(\mathbf{p})$ дискретного ансамбля с распределением вероятностей $\mathbf{p}$ является выпуклой $\cap$ функцией аргумента $\mathbf{p}$.

Доказательство. Прежде всего, заметим, что постановка вопроса о выпуклости энтропии корректна, поскольку ее область определения выпукла в соответствии с утверждением 1, доказанным выше. В соответствии с определением энтропии

$$H(\mathbf{p}) = -\sum_{m=1}^{M} p_m \log p_m = \sum_{m=1}^{M} f_m(\mathbf{p}). \qquad (1.4.4)$$

Рассмотрим слагаемые $f_m(\mathbf{p})$. Каждое из них представляет собой функцию одной переменной. Вторая производная этой функции имеет вид $-1/p_m$ и, следовательно, отрицательна при всех $p_m \in (0,1)$. Таким образом, энтропия выпукла в силу свойства 4. Тем самым теорема доказана.■

Заметим, что слагаемые $f_m(\mathbf{p})$ в (1.4.4) – строго выпуклые функции. Следовательно, энтропия – также строго выпукла.

В заключение параграфа мы приведем пример применения свойств выпуклых функций как инструмента исследования свойств информационных мер. Докажем оставшееся недоказанным 6-е свойство энтропии.

Доказательство свойства 1.3.6. Пусть имеется ансамбль X, $|X| = M$. Запишем вероятности букв в виде вектора $\mathbf{p} = (p_1,..., p_M)$. Для энтропии ансамбля вместо $H(X)$ будем использовать обозначение $H(\mathbf{p})$. Докажем, что частный случай утверждения, а именно, докажем, что выравнивание вероятностей первого и второго элементов множества X приводит к увеличению энтропии. Обобщение на случай произвольного подмножества $A \subseteq X$ оставим читателю в качестве упражнения. Обозначим $\tilde{\mathbf{p}} = \big((p_1 + p_2)/2, (p_1 + p_2)/2, p_3,..., p_M\big)$. Нужно доказать неравенство

$$H(\mathbf{p}) \le H(\tilde{\mathbf{p}}). \qquad (1.4.5)$$

Для этого введем обозначения

$$\mathbf{p}' = \mathbf{p} = (p_1, p_2, p_3,...p_M);$$
$$\mathbf{p}'' = (p_2, p_1, p_3,...p_M).$$

Справедливо тождество $\tilde{\mathbf{p}} = (\mathbf{p}' + \mathbf{p}'')/2$. Из выпуклости энтропии следует, что

$$0{,}5H(\mathbf{p}') + 0{,}5H(\mathbf{p}'') \le H(\tilde{\mathbf{p}}).$$

Из свойства 1.3.3 энтропии имеем $H(\mathbf{p}') = H(\mathbf{p}'') = H(\mathbf{p})$. Поэтому из последнего неравенства следует (1.4.5).■

1.5. Условная энтропия

Как уже отмечалось, для эффективного кодирования информации необходимо учитывать статистическую зависимость сообщений. Наша ближайшая цель – научиться подсчитывать информационные характеристики последовательностей зависимых сообщений. Начнем с двух сообщений.

Рассмотрим ансамбли $X = \{x\}$ и $Y = \{y\}$ и их произведение $XY = \big\{(x, y), p(x, y)\big\}$. Для любого фиксированного $y \in Y$ можно построить условное распределение вероятностей $p(x \mid y)$ на множестве X и для каждого $x \in X$ подсчитать собственную информацию

$$I(x \mid y) = -\log p(x \mid y),$$

которую называют *условной собственной информацией* сообщения x при фиксированном y.

Ранее мы назвали энтропией ансамбля X среднюю информацию сообщений $x \in X$. Аналогично, усреднив условную информацию $I(x \mid y)$ по $x \in X$, получим величину

$$H(X \mid y) = -\sum_{x \in X} p(x \mid y) \log p(x \mid y), \qquad (1.5.1)$$

называемую условной энтропией X при фиксированном $y \in Y$. Заметим, что в определении (1.5.1) имеет место неопределенность в случае, когда $p(x \mid y) = 0$. В п. 1.3 отмечалось, что выражение вида $z \log z$ стремится к нулю при $z \to 0$ и на этом основании мы считали слагаемые энтропии, соответствующие буквам x с вероятностью $p(x) = 0$, равными нулю. Точно также в (1.5.1) мы считаем равными нулю слагаемые, для которых $p(x \mid y) = 0$.

Вновь введенная энтропия $H(X \mid y)$ – случайная величина, поскольку она зависит от случайной переменной y. Чтобы получить неслучайную информационную характеристику пары вероятностных ансамблей, нужно выполнить усреднение в (1.5.1) по всем значениям y. Величина

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

называется *условной энтропией* ансамбля X при фиксированном ансамбле Y. Отметим ряд свойств условной энтропии.

Свойство 1.5.1. $H(X \mid Y) \geq 0.$

Свойство 1.5.2. $H(X \mid Y) \leq H(X)$, причем равенство имеет место в том и только в том случае, когда ансамбли X и Y независимы.

Свойство 1.5.3. $H(XY) = H(X) + H(Y \mid X) = H(Y) + H(X \mid Y).$

Свойство 1.5.4.

$H(X_1 ... X_n) = H(X_1) + H(X_2 \mid X_1) + H(X_3 \mid X_1 X_2) + ... + H(X_n \mid X_1, ..., X_{n-1})$.

Свойство 1.5.5. $H(X \mid YZ) \leq H(X \mid Y)$ причем равенство имеет место в том и только в том случае, когда ансамбли X и Y условно независимы при всех $z \in Z$.

Свойство 1.5.1 доказывается так же как свойство энтропии 1.3.1.

Мы приведем два доказательства свойства 1.5.2. Первое основано на относительно простых выкладках и использует неравенство (1.3.1). Другое доказательство дается как пример использования свойства выпуклости энтропии.

Доказательство свойства 1.5.2. По аналогии с доказательством свойства энтропии 1.3.2:

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

Это же свойство можно доказать иначе**.** Поскольку энтропия ансамбля зависит только от распределения вероятностей, вместо выражения $H(X)$ будем писать $H(\mathbf{p}_X)$, где $\mathbf{p}_X$ означает записанное в виде вектора распределение вероятностей. По определению условной энтропии

$$H(X \mid Y) = \mathbf{M}_Y \left[H(\mathbf{p}_{X|y}) \right] \leq H\left(\mathbf{M}_Y \left[\mathbf{p}_{X|y} \right] \right) = H\left(\mathbf{p}_X \right) = H(X),$$

где $\mathbf{p}_{X|y}$ – условное распределение на X при фиксированном y, неравенство основано на выпуклости энтропии, следующее затем равенство основано на формуле полной вероятности:

$$\mathbf{M}_y \left[p(x \mid y) \right] = \sum_y p(x \mid y) p(y) = p(x) . \blacksquare$$

Доказательство свойств 1.5.3 и 1.5.4. По формуле умножения вероятностей

$$p(x, y) = p(x) p(y \mid x) = p(y) p(x \mid y),$$

$$p(x_1, ..., x_n) = p(x_1) p(x_2 \mid x_1) ... p(x_n \mid x_1, ..., x_{n-1}).$$

Обе стороны этих тождеств нужно прологарифмировать, а затем усреднить по всем случайным переменным. ■

Доказательство свойства 1.5.5. Рассмотрим ансамбль $XYZ = \{(x, y, z), p(x, y, z)\}$. При каждом $z \in Z$ определены условные распределения $p(x, y \mid z)$ и $p(x \mid z)$. Для этих распределений, запишем энтропии

$$H(X \mid Y, z) = \mathbf{M}_{XY|z} [-\log p(x \mid yz)],$$

$$H(X \mid z) = \mathbf{M}_{X|z} [-\log p(x \mid z)].$$

По свойству 1.5.2

$$H(X \mid Y, z) \leq H(X \mid z).$$

После усреднения по всем z получим требуемое неравенство. ■

Обсудим «физический смысл» сформулированных свойств условной энтропии. Свойство 1.5.2 устанавливает, что условная энтропия ансамбля не превышает его безусловной энтропии. Свойство 1.5.5 усиливает это утверждение. Из него следует, что условная энтропия не увеличивается с увеличением числа условий. Оба эти факта неудивительны, они отражают тот факт, что дополнительная информация об ансамбле X, содержащаяся в сообщениях других ансамблей, *в среднем,* уменьшает информативность (неопределенность) ансамбля X. Замечание «*в среднем»* здесь очень важно, поскольку неравенство

$$H(X \mid y) \le H(X),$$

вообще говоря, неверно.

Из свойств 1.5.1 – 1.5.5 следует неравенство

$$H(X_1 ... X_n) \le \sum_{i=1}^{n} H(X_i), \tag{1.5.2}$$

в котором равенство возможно только в случае совместной независимости ансамблей $X_1, \ldots, X_n$.

Применим выявленные свойства условной энтропии для доказательства свойства энтропии 1.3.7, устанавливающего невозрастание энтропии при обработке информации.

Доказательство свойства 1.3.7. Имеем ансамбль $X = \{x, p(x)\}$, определенную на нем функцию $g(x)$ и ансамбль $Y = \{y = g(x)\}$. Нужно доказать, что

$$H(Y) \le H(X) \ . \tag{1.5.3}$$

По свойствам энтропии

$$\begin{aligned} H(XY) &= H(X \mid Y) + H(Y) = \\ &= H(Y \mid X) + H(X). \end{aligned} \tag{1.5.4}$$

Поскольку значения $g(x)$ однозначно определены при заданном x, имеем $H(Y \mid X) = 0$. В то же время $H(X \mid Y) \ge 0$. С учетом этих обстоятельств из (1.5.4) следует (1.5.3). ■

Напомним, что вычисление энтропии – это вычисление затрат на передачу или хранение букв источника. Свойства условной энтропии подсказывают, что при передаче буквы X_{n+1} следует использовать то обстоятельство, что предыдущие буквы $X_1, \ldots, X_n$ уже известны на приемной стороне. Это позволит вместо $H(X_{n+1})$ бит потратить меньшее количество $H(X_{n+1} \mid X_1, ..., X_n)$ бит. В то же время неравенство (1.5.2) указывает другой подход к экономному кодированию. Из этого неравенства следует, что буквы перед кодированием нужно объединять в блоки и эти блоки рассматривать как буквы нового «расширенного» источника. Затраты будут меньше, чем при независимом кодировании букв. Какой из двух подходов эффективнее?

Ниже мы дадим более точную количественную характеристику этих двух подходов, но перед этим нам нужно вспомнить некоторые определения из теории вероятностей.

1.6. Дискретные случайные последовательности. Цепи Маркова

Вместо отдельных ансамблей и произведений конечного числа ансамблей мы будем рассматривать теперь случайные последовательности из произвольного числа событий. Если элементы случайной последовательности – вещественные числа, то такие последовательности называются *случайными процессами.*

Номер элемента в последовательности трактуется как момент времени, в который появилось данное значение. Вообще говоря, множество значений времени может быть непрерывным либо дискретным, множество значений случайной последовательности

также может быть непрерывным либо дискретным. Сейчас нас интересуют только дискретные процессы дискретного времени.

Случайный процесс $x_1, x_2, \ldots$ со значениями $x_i \in X$, $i = 1,2,...$ *задан,* если для любых n указан способ вычисления совместных распределений вероятностей $p(x_1,...,x_n)$.

Проще всего задать случайный процесс, предположив, что его значения в различные моменты времени независимы и одинаково распределены. Тогда

$$p(x_1,...,x_n) = \prod_{i=1}^{n} p(x_i),$$

где $p(x_i)$ – вероятность появления $x_i \in X$ в момент времени i. Для описания такого процесса достаточно указать вероятности $p(x)$ для всех $x \in X$ (всего $|X|-1$ вероятностей).

Описание более сложных моделей процессов будет громоздким, если не сделать упрощающих предположений. Нам не обойтись без предположения о стационарности.

Процесс называется *стационарным,* если для любых n и t имеет место равенство

$$p(x_1,...,x_n) = p(x_{1+t},...,x_{n+t}),$$

в котором подразумевается, что $x_i = x_{i+t}$, $i = 1,...,n$.

Иными словами, случайный процесс стационарен, если вероятность любой последовательности не изменяется при ее сдвиге во времени (не зависит от положения последовательности на оси времени).

Числовые характеристики, в частности, математическое ожидание стационарных процессов не зависят времени. Рассматривая стационарные процессы, мы сможем вычислять независящие от времени информационные характеристики случайных процессов.

Мы уже рассмотрели один пример стационарного процесса – процесс, значения которого независимы и одинаково распределены. Источник, порождающий такой процесс, называют *дискретным постоянным источником* (ДПИ).

Простейшей моделью источника, порождающего зависимые сообщения, является марковский источник.

Случайный процесс $x_1, x_2, ...$ называют *цепью Маркова связности* s, если для любых n и для любых $\mathbf{x} = (x_1,...,x_n) \in X^n$ справедливы соотношения

$$p(\mathbf{x}) = p(x_1,...,x_s)\,p(x_{s+1} \mid x_1,...,x_s)\,p(x_{s+2} \mid x_2,...,x_{s+1})...p(x_n \mid x_{n-s},...,x_{n-1}).$$

Иными словами, мы называем марковским процессом связности s такой процесс, для которого при $n > s$

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

то есть условная вероятность текущего значения при известных s предшествующих не зависит от всех других предшествующих значениях.

Описание марковского процесса задается начальным распределением вероятностей на последовательностях из первых s значений и условными вероятностями вида $p(x_n \mid x_{n-s},...,x_{n-1})$ для всевозможных последовательностей $(x_{n-s},...,x_n)$. Если указанные условные вероятности не изменяются при сдвиге последовательностей $(x_{n-s},...,x_n)$ во времени, марковская цепь называется *однородной.*

Однородная марковская цепь связности $s = 1$ называется *простой* цепью Маркова. Для описания простой цепи Маркова с множеством состояний $X = \{0,1,...,M-1\}$ достаточно указать начальное распределение вероятностей $\{p(x_1), x_1 \in X\}$ и условные вероятности

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

называемые переходными вероятностями цепи Маркова.

Переходные вероятности удобно записывать в виде квадратной матрицы размерности $M \times M$:

$$\Pi = \begin{bmatrix} \pi_{00} & \dots & \pi_{0,M-1} \\ \dots & \dots & \dots \\ \pi_{M-1,0} & \dots & \pi_{M-1,M-1} \end{bmatrix},$$

называемой **матрицей переходных вероятностей**. Эта матрица – стохастическая (неотрицательная, сумма элементов каждой строки равна 1).

Обозначим через $\mathbf{p}_t$ стохастический вектор, компоненты которого – вероятности состояний цепи Маркова в момент времени t, то есть $\mathbf{p}_t = \left(p_t(0),..., p_t(M-1)\right)$, где $p_t(i)$ есть вероятность состояния i в момент времени t, $i = 0,...,M-1$. Из формулы полной вероятности следует, что

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

или в матричной форме

$$\mathbf{p}_{t+1} = \mathbf{p}_t \Pi . \qquad (1.6.1)$$

Отсюда для произвольного числа шагов n получаем

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

Значит, вероятности перехода за n шагов могут быть вычислены как элементы матрицы Π^n. Из этих (1.6.1) видим, что распределение вероятностей в момент времени t зависит от величины t и от начального распределения $\mathbf{p}_1$. Отсюда следует, что в общем случае рассматриваемый случайный процесс нестационарен. Предположим, однако, что существует стохастический вектор $\mathbf{p}$, удовлетворяющий уравнению

$$\mathbf{p} = \mathbf{p}\,\Pi. \qquad (1.6.2)$$

Положим $\mathbf{p}_1 = \mathbf{p}$. Тогда, воспользовавшись (1.6.1), получим $\mathbf{p}_2 = \mathbf{p}$ и, в конечном итоге, $\mathbf{p}_t = \mathbf{p}$ при всех t. Таким образом, однородная марковская цепь стационарна, если в качестве начального распределения выбрано решение уравнения (1.6.2).

Стохастический вектор $\mathbf{p}$, удовлетворяющий уравнению (1.6.2), называется **стационарным распределением** для цепи Маркова, задаваемой матрицей переходных вероятностей Π.

Финальным распределением вероятностей называют вектор

$$\mathbf{p}_\infty = \lim_{t\to\infty} \mathbf{p}_\mathbf{t} = \lim_{t\to\infty} \mathbf{p}_1 \Pi^t , \qquad (1.6.3)$$

(если предел существует).

Из этого определения следует, что финальное распределение – распределение вероятностей в момент времени t бесконечно далекий от начального момента времени $t = 1$. Было бы естественно ожидать, что оно не зависит от начального распределения $\mathbf{p}_1$. Оно не зависит также и от времени. Таким образом, распределение $\mathbf{p}_\infty$ тоже (как и $\mathbf{p}$), в некотором смысле, – стационарное распределение. Как же соотносятся между собой $\mathbf{p}_\infty$ и $\mathbf{p}$?

Оказывается, для широкого класса простых цепей Маркова предел в (1.6.3) не зависит от начального распределения $\mathbf{p}_1$ и равен единственному решению уравнения (1.6.2), то есть $\mathbf{p}_\infty = \mathbf{p}$. Такие цепи называют эргодическими.

Как определить по матрице Π эргодична ли соответствующая цепь Маркова? Ответ заведомо положительный, если все элементы матрицы Π положительны (не равны нулю). Более точное (но и сложнее проверяемое) условие состоит в том, что должна

существовать некоторая положительная степень n_0 матрицы П такая, что все элементы матрицы Π^n положительны при любых $n \geq n_0$.

Чтобы сформулировать необходимое и достаточное условие эргодичности, придется ввести несколько определений.

Состояние цепи i *достижимо* из состояния j, если для некоторого n вероятность перехода из состояния j в состояние i за n шагов положительна. Множество состояний C называется *замкнутым*, если никакое состояние вне C не может быть достигнуто из состояния, входящего в C.

Цепь называется *неприводимой*, если в ней нет никаких замкнутых множеств кроме множества всех состояний. Цепь Маркова неприводима тогда и только тогда, когда все ее состояния достижимы друг из друга.

Состояние i называется *периодическим,* если существует такое $t>1$, что вероятность перехода из i в i за n шагов равна нулю при всех n не кратных t. Цепь, не содержащая периодических состояний, называется *непериодической*.

Непериодическая неприводимая цепь Маркова эргодична.

1.7. Энтропия на сообщение дискретного стационарного источника

Рассмотрим произвольный дискретный стационарный источник, порождающий последовательность $(x_1, x_2, ..., x_t, ...)$, $x_t \in X_t = X$. Из предположения о стационарности следует, что распределение вероятностей для буквы x_t, порождаемой в момент времени t не зависит от t. Следовательно, величина энтропии этого распределения $H(X_n) = H(X)$ также не зависит от времени. Назовем ее *одномерной энтропией источника* (или соответствующего случайного процесса). Обозначим ее как $H_1(X)$.

Как уже было отмечено, величина $H_1(X)$ не определяет полностью информационные характеристики процесса, поскольку не учитывает зависимости букв.

Рассмотрим последовательность из n последовательных букв источника $\mathbf{x} = (x_1, ..., x_n) \in X_1 X_2 ... X_n = X^n$. Для стационарного процесса энтропия распределения вероятностей на таких блоках $H(X_1 ... X_n) = H(X^n)$ не зависит от расположения блока во времени, ее называют n*-мерной энтропией* источника.

Величина $H(X^n)$ определяет среднее количество информации в последовательности из n букв. Нормированную величину

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

называют *энтропией на букву последовательности длины* n. Интуиция подсказывает, что значения $H_n(X)$ при больших n могли бы служить адекватной мерой информативности источника.

Другой подход к измерению информации, порождаемой произвольным стационарным источником, состоит в том, что при передаче буквы x_n все предыдущие буквы $x_1, ..., x_{n-1}$ можно считать известными декодеру. Среднее количество подлежащей передаче информации об x_n определяется величиной условной энтропии $H(X_n \mid X_1, ..., X_{n-1})$. В силу стационарности конкретные значения индексов не играют роли, важна лишь длина предыстории. Поэтому используется следующее обозначение:

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

Следующая теорема устанавливает некоторые свойства двух информационных мер стационарных источников.

Теорема 1.7.1. Для дискретного стационарного источника

A. $H(X \mid X^n)$ не возрастает с увеличением n;

B. $H_n(X)$ не возрастает с увеличением n;

C. $H_n(X) \geq H(X / X^{n-1})$;

D. $\lim\limits_{n\to\infty} H_n(X) = \lim\limits_{n\to\infty} H(X \mid X^n)$.

Доказательство. Утверждение A следует из невозрастания энтропии с увеличением числа условий (свойство 1.5.5).

Из свойства 1.5.4 и стационарности источника имеем

$$H(X^n) = H(X) + H(X \mid X^1) + \ldots + H(X \mid X^{n-1}).$$

Заметим, что в правой части n слагаемых, из которых последнее – наименьшее (в силу свойств 1.5.2 и 1.5.5). Поделив обе части на n, убеждаемся в справедливости утверждения C.

Чтобы убедиться в справедливости утверждения B, выполним преобразования

$$H(X^{n+1}) = H(X_1 \ldots X_n X_{n+1}) = H(X_1 \ldots X_n) + H(X_{n+1} \mid X_1, \ldots, X_n) =$$
$$= H(X^n) + H(X \mid X^n) \leq H(X^n) + H(X \mid X^{n-1}) \leq$$
$$\leq H(X^n) + H_n(X) = (n+1) H_n(X).$$

В этой цепочке первые два перехода основаны на свойствах энтропии, затем использована стационарность. Первое неравенство опирается на уже доказанное утверждение A, следующее неравенство – на доказанное утверждение C. Далее использовано определение величины $H_n(X)$. Если теперь поделить правую и левую часть на $(n+1)$, убеждаемся в справедливости утверждения B.

Перейдем к доказательству последнего утверждения. Прежде всего, отметим, что последовательности $H_n(X)$ и $H(X \mid X^n)$ не возрастают и ограничены снизу (поскольку неотрицательны). Значит, оба предела существуют. Из свойства C следует, что

$$\lim_{n\to\infty} H_n(X) \geq \lim_{n\to\infty} H(X \mid X^n). \qquad (1.7.1)$$

В то же время, при любых натуральных $m < n$ имеют место соотношения

$$H(X^n) = H(X_1 \ldots X_n) = H(X_1 \ldots X_m) + H(X_{m+1} \ldots X_n \mid X_1, \ldots, X_m) =$$
$$= m H_m(X) + H(X_{m+1} \mid X_1, \ldots, X_m) + \ldots + H(X_n \mid X_1, \ldots, X_{n-1}) \leq$$
$$\leq m H_m(X) + (n-m) H(X \mid X^m).$$

Здесь использованы свойства условной энтропии, стационарность источника и невозрастание последовательности $H(X \mid X^m)$ с увеличением числа условий m. Поделим обе части полученного неравенства на n и перейдем к пределу при $n \to \infty$. Получим неравенство

$$\lim_{n\to\infty} H_n(X) \leq H(X \mid X^m),$$

справедливое при любых m. Устремляя m к бесконечности, получаем

$$\lim_{n\to\infty} H_n(X) \leq \lim_{m\to\infty} H(X \mid X^m). \qquad (1.7.2)$$

Из (1.7.1) и (1.7.2) вытекает доказываемое утверждение. ■

Введем обозначения

$$H_\infty(X) = \lim_{n\to\infty} H_n(X), \quad H(X \mid X^\infty) = \lim_{n\to\infty} H(X \mid X^n).$$

Основной результат теоремы состоит в том, что $H_\infty(X) = H(X \mid X^\infty)$. В дальнейшем, изучая конструктивные методы кодирования, мы убедимся в том: что именно эта

величина определяет минимально возможные удельные затраты бит на передачу одной буквы стационарного источника.

По сути, мы рассмотрели два подхода к анализу информативности стационарного источника:

- введение расширенного алфавита, буквами которого служат блоки из n символов источника;
- учет зависимости текущей буквы от n предшествующих букв.

Выяснилось, что, хотя при каждом конкретном n второй подход дает более оптимистические оценки затрат на передачу или хранение информации (свойство С). В пределе с увеличением параметра n, подходы становятся эквивалентными.

1.8. Примеры вычисления энтропии на сообщение

В предыдущем параграфе формально доказано существования пределов

$$\lim_{n\to\infty} H_n(X) = H_\infty(X),\ \lim_{n\to\infty} H(X \mid X^n) = H(X \mid X^\infty)$$

и тождество

$$H_\infty(X) = H(X \mid X^\infty).$$

В этом параграфе мы изучим поведение последовательностей $H_n(X)$ и $H(X \mid X^n)$ для простых моделей дискретных постоянных источников. Возможно, это сделает более понятными и естественными сформулированные выше фундаментальные утверждения.

Пример 1.8.1. Дискретный постоянный источник.

Дискретным постоянным источником мы назвали дискретный стационарный источник без памяти. По свойствам энтропии для источника без памяти имеем

$$H(X_1 \ldots X_n) = H(X_1) + \ldots + H(X_n).$$

Учитывая стационарность, перепишем это тождество в виде

$$H(X^n) = nH(X).$$

Поделив обе стороны тождества на n, получим, что при всех n справедливо равенство

$$H_n(X) = H(X).$$

Отсюда следует, что

$$H_\infty(X) = H(X).$$

Таким образом, энтропия на сообщение дискретного постоянного источника в точности равна его одномерной энтропии.

К тому же результату можно было прийти с другой стороны. В силу стационарности и отсутствия зависимости между сообщениями

$$H(X \mid X^n) = H(X_{n+1} \mid X_1, \ldots, X_n) = H(X),$$

следовательно,

$$H(X \mid X^\infty) = H(X).$$

Таким образом, для рассматриваемой модели анализ информационных характеристик сводится к подсчету энтропии одномерного распределения. Отсюда совсем не следует, что при кодировании источников независимых сообщений нужно кодировать каждую букву независимо от других.

Пример 1.8.2. Марковский источник.

Для стационарного источника, описываемого моделью цепи Маркова связности s, можем записать

$$H(X \mid X^n) = H(X_{n+1} \mid X_1, \ldots, X_n) = H(X_{n+1} \mid X_{n-s+1}, \ldots, X_n) = H(X \mid X^s).$$

Правая часть не зависит от n, значит

$$H(X \mid X^\infty) = H(X \mid X^s).$$

Рассмотрим другой подход. Используя стационарность и свойства условной энтропии, запишем

$$H(X^n) = H(X_1...X_sX_{s+1}...X_n) = H(X_1...X_s) + H(X_{s+1}...X_n \mid X_1,...,X_s). \qquad (1.8.1)$$

Для второго слагаемого имеем

$$H(X_{s+1}...X_n \mid X_1,...,X_s) = H(X_{s+1} \mid X_1,...,X_s) + H(X_{s+2} \mid X_1,...,X_s,X_{s+1}) + ... + H(X_n \mid X_1,...,X_{n-1})$$

Учитывая марковость и стационарность, получаем

$$H(X_{s+1}...X_n \mid X_1,...,X_s) = (n-s)H(X \mid X^s).$$

Теперь (1.8.1) принимает вид

$$H(X^n) = sH_s(X) + (n-s)H(X \mid X^s). \qquad (1.8.2)$$

Теперь можно поделить обе части тождества на n и устремить n к бесконечности. Результатом будет требуемое равенство

$$H_\infty(X) = H(X \mid X^s).$$

Поучительно записать (1.8.2) в виде

$$H(X^n) = nH(X \mid X^s) + s(H_s(X) - H(X \mid X^s)) =$$
$$= nH(X \mid X^n) + s(H_s(X) - H(X \mid X^s)).$$

Отсюда, в частности, хорошо видно, какова разница между $H_n(X)$ и $H(X \mid X^n)$. При разбиении последовательности букв на блоки длины n мы получаем среднюю энтропию на сообщение немного большую, чем минимальная достижимая величина $H(X \mid X^s)$. «Потери» или дополнительные затраты связаны с «забвением» тех букв, которые предшествовали рассматриваемому блоку.

При кодировании марковского источника предел энтропии на сообщение подсчитывается по (s+1)-мерному распределению вероятностей.

1.9. Постановка задачи равномерного кодирования дискретного источника

Теперь, когда мы знаем теоретическую характеристику информационной производительности источника, можно было бы перейти к решению практической задачи сжатия информации. Однако в данном параграфе мы рассмотрим метод равномерного кодирования, который не имеет почти никакого практического значения. Более того, он, по сути, не является сжатием информации без потерь. Тем не менее, изучение равномерного кодирования представляется нужным по двум причинам. Во-первых, станет еще понятнее, почему энтропия совпадает с предельно достижимой скоростью передачи (хранения) информации (эта скорость называется скоростью создания информации источником). Во-вторых, мы на этом примере познакомимся с традиционным для теории информации способом формулировки теоретических результатов – доказательством прямой и обратной теорем кодирования.

При равномерном кодировании последовательность порождаемых источником сообщений $x_1, x_2, ...$, $x_i \in X$, $i = 1,2,...$ разбивается на блоки одинаковой длины N. Каждый такой блок кодируется независимо от других блоков.

Для кодирования используется некоторый алфавит $A = \{a\}$, называемым кодовым, элементы алфавита называют кодовыми символами. Мы ограничимся рассмотрением двоичных кодов, то есть кодов над алфавитом $A = \{0,1\}$.

Кодом длины n называется любое подмножество C множества A^n, то есть любое подмножество множества последовательностей длины n. Элементы кода C называют кодовыми словами. Мощность кода $|C|$ – это количество кодовых слов в коде C.

Скоростью равномерного кода называется величина

$$R = \frac{\log |C|}{N} \text{ (бит/символ источника).} \qquad (1.9.1)$$

Смысл такого определения скорости кодирования станет понятнее, если мы предположим, что кодом служит все множество последовательностей длины n, то есть $C = A^n$. При этом скорость кода равна

$$R = \frac{n}{N} \text{ (бит/символ источника).}$$

Если работа кодера состоит в том, что каждый блок из N символов источника заменяется на n двоичных символов, записываемых затем в запоминающее устройство или передаваемых по каналу связи, то смысл определения скорости прозрачен. Скорость кода – это удельные затраты на передачу символа источника.

Если же $C \subset A^n$, то величина $\log |C|$ представляет собой количество бит необходимое для указания номера кодового слова, а скорость кода – количество бит, затрачиваемых на передачу одной буквы источника.

Работа кодера (алгоритм кодирования) описывается отображением множества X^N на множество слов кода C. Декодирование задается отображением C на X^N. Если оба отображения взаимно-однозначные, то на выходе декодера можно будет получить точную копию передаваемой последовательности. Взаимно-однозначное кодирование возможно только тогда, когда

$$|X|^N \le |C| \qquad (1.9.2)$$

или

$$R \ge \log |X| \ge H(X).$$

Следовательно, если буквы источника не равновероятны ($H(X) < \log |X|$), то скорость кода окажется заведомо больше энтропии источника.

Рассмотрим теперь ситуацию, когда условие (1.9.2) не выполнено. Тогда кодовых слов недостаточно для того, чтобы сопоставить каждой последовательности источника свое кодовое слово. Для некоторых последовательностей сообщений не найдется кодового слова, по которому декодер смог бы однозначно восстановить переданную информацию. В общем виде процесс кодирования и декодирования можно описать следующим образом.

Выделим в X^N подмножество T, такое, что $|T| = |C|$, и каждой последовательности из множества T сопоставим индивидуальное кодовое слово. Множество T мы будем называть множеством однозначно кодируемых последовательностей. Остальным последовательностям из X^N сопоставим произвольные кодовые слова (например, всем последовательностям из T^c сопоставим одно и то же кодовое слово). Декодер при получении некоторого кодового слова $\mathbf{c} \in C$ будет выдавать получателю соответствующую этому слову последовательность из множества T.

Из описания следует, что каждый раз, когда источник породит последовательность из дополнения к множеству T, выход декодера не будет совпадать со входом кодера. Это событие называют ошибкой кодирования и вероятность

$$P_e = P(\mathbf{x} \notin T)$$

называется вероятностью ошибки кодирования.

Пример. Рассмотрим троичный постоянный источник $X = \{a, b, c\}$ с распределением вероятностей $p(a) = 1/2$, $p(b) = 1/3$, $p(c) = 1/6$. Положим $N = 2$. В этом случае множество X^N состоит из 9 пар. Распределение вероятностей и пример двоичного кода длины $n = 3$ приведены в Таблице 1.9.1. Кодом служит множество $C = \{0,1\}^3$ (множество всех последовательностей длины 3). Скорость кода равна $R = 3/2$ бита на букву источника. Из таблицы ясно, что множество T имеет вид

$T = \{aa, ab, ac, ba, bb, bc, ca, cb\}$. Вероятность ошибки $P_e = p(cc) = 1/36$. При появлении на выходе источника последовательности cc декодер будет выдавать получателю последовательность cb.

Таблица 1.9.1

Пример равномерного кода

Последовательность источника	Вероятность	Кодовое слово
aa	1/4	000
ab	1/6	001
ac	1/12	010
ba	1/6	011
bb	1/9	100
bc	1/18	101
ca	1/12	110
cb	1/18	111
cc	1/36	111

Отметим, что в данном примере кодирование могло быть и другим, но при любом другом выборе множества T вероятность ошибки была не меньше 1/36. В то же время, выбор кодового слова для последовательности из дополнения к T не влияет на вероятность ошибки. ■

Итак, задача построения равномерного кода со скоростью R для последовательностей длины n эквивалентна задаче выбора 2^{nR} однозначно кодируемых сообщений. Хотелось бы построить такой код, который обеспечивал бы одновременно малую скорость и малую вероятность ошибки. Поскольку скорость может быть уменьшена только за счет уменьшения множества T, с уменьшением скорости неизбежно увеличивается вероятность ошибки. Представляет интерес ответ на вопрос: с какой скоростью возможно кодирование источника при пренебрежимо малой вероятности ошибки?

Для заданного стационарного источника число H называется *скоростью создания информации*, если для любого $R > H$ существует равномерный код со скоростью R, обеспечивающий сколь угодно малую вероятность ошибки, и, в то же время, при любой скорости кода $R < H$ вероятность ошибки не может быть сделана меньше некоторой положительной величины ε.

Для того, чтобы утверждать, что константа H является скоростью создания информации для данной модели источника, нужно доказать два утверждения. Первое устанавливает, что при $R > H$ достижима сколь угодно малая вероятность ошибки. Это утверждение называют *прямой теоремой кодирования***.** Второе утверждение состоит в том, что при $R < H$ вероятность ошибки не может быть сделана произвольно малой. Это утверждение называется *обратной теоремой кодирования***.**

Мы покажем, что для стационарных источников скорость создания информации совпадает с энтропией на сообщение источника, т. е. что имеют место равенства

$$H = H(X \mid X^{\infty}) = H_{\infty}(X).$$

В определении понятия «скорость создания информации источником» неявно участвует ограничение на множество способов кодирования, а именно, предполагается, что блоки из фиксированного числа сообщений n преобразуются в кодовые слова фиксированной длины N. Для обозначения кодов такого типа используют аббревиатуру FF (fixed-to-fixed). Помимо FF-кодов можно рассматривать VF-коды (variable-to-fixed), преобразующие блоки переменной длины в кодовые слова фиксированной длины, FV-коды (fixed-to-variable), когда для кодирования блоков фиксированной длины используются коды с переменной длиной кодовых слов, и наконец самый широкий класс

кодов – VV-коды (variable-to-variable), допускающие разбиение сообщений на блоки переменной длины и кодирование блоков неравномерными кодами. Строго говоря, чтобы утверждать, что число H – скорость создания информации источником, при доказательстве обратной теоремы кодирования нужно допустить выбор способов кодирования из любого из четырех перечисленных множеств. Ниже в п. 1.12 мы приведем доказательство обратной теоремы для FF-кодов. В части 2 будут изучаться FV-коды и доказана соответствующая теорема кодирования. Доказательство теорем для двух других классов выходит за рамки нашего курса.

1.10. Неравенство Чебышева. Закон больших чисел

Доказательство теорем кодирования опирается на оценки вероятностей уклонения случайных величин от средних значений. В этом параграфе мы приведем необходимые сведения из теории вероятностей.

Точный подсчет вероятностей больших уклонений бывает невозможен либо в силу сложности задачи, либо по причине отсутствия точных сведений о вероятностной модели. Поэтом часто пользуются оценками, верхними и нижними. Наиболее важными с практической точки зрения являются верхние оценки, гарантирующие, что вероятность неудачи не превосходит некоторого значения.

Рассмотрим дискретную случайную величину (с. в.) $X = \{x, p(x)\}$. Предположим что все ее значения $x \in X$ неотрицательны и при этом предположении оценим вероятность события $P(x \ge A)$ для некоторого числа $A > 0$.

Имеем

$$P(x \ge A) = \sum_{x \ge A} p(x) \le \sum_{x \ge A} \frac{x}{A} p(x) \le \frac{1}{A} \sum_{x \in X} x p(x) = \frac{\mathbf{M}[x]}{A}.$$

Первое из двух неравенств основано на том, что в области суммирования $x / A \ge 1$. Второе неравенство справедливо потому, что, расширив область суммирования на все множество X, мы добавили к сумме только неотрицательные слагаемые (все значения x неотрицательны).

Введем обозначение $m_x = \mathbf{M}[x]$. Полученный результат запишем в виде

$$P(x \ge A) \le \frac{m_x}{A}. \qquad (1.10.1)$$

Ценность этой формулы состоит в том, что для оценки искомой вероятности достаточно знать только одну числовую характеристику с. в. – математическое ожидание.

Пусть теперь $X = \{x, p(x)\}$ – произвольная (необязательно неотрицательная) с. в.. Для произвольного $\varepsilon > 0$ оценим вероятность $P(|x - m_x| \ge \varepsilon)$, отклонения с. в. от ее среднего значения на величину не меньшую чем ε. Положим $y = |x - m_x|$. Для этой неотрицательной с.в. используя (1.10.1), получаем

$$P(y \ge \varepsilon) = P(y^2 \ge \varepsilon^2) \le \frac{\mathbf{M}[y^2]}{\varepsilon^2} = \frac{\mathbf{M}[(x - m_x)^2]}{\varepsilon^2} = \frac{\mathbf{D}[x]}{\varepsilon^2}.$$

Результат запишем в виде

$$P(|x - m_x| \ge \varepsilon) \le \frac{\sigma_x^2}{\varepsilon^2}, \qquad (1.10.2)$$

где $\sigma_x^2 = \mathbf{D}[x]$. Неравенство (1.10.2) называется неравенством Чебышева.

Рассмотрим теперь последовательность из n независимых с. в. X_i, $i = 1,...,n$, имеющих одинаковое математическое ожидание m_x и одинаковую дисперсию σ_x^2. Нас интересует вероятность

$$P\left(\left|\frac{1}{n}\sum_{i=1}^{n}x_i - m_x\right| \geq \varepsilon\right)$$

отклонения среднего арифметического n с. в. от их математического ожидания на величину не меньшую ε. Положим $y = \frac{1}{n}\sum_{i=1}^{n}x_i$. Из свойств математического ожидания и дисперсии, принимая во внимание независимость с.в. $x_1,...,x_n$, имеем

$$\mathbf{M}[y] = m_x,\ \mathbf{D}[y] = \frac{1}{n}\sigma_x^2.$$

Применив к с.в. y неравенство Чебышева (1.10.2), получаем новое важное неравенство

$$P\left(\left|\frac{1}{n}\sum_{i=1}^{n}x_i - m_x\right| \geq \varepsilon\right) \leq \frac{\sigma_x^2}{n\varepsilon^2}\ , \qquad (1.10.3)$$

это неравенство Чебышева для суммы независимых случайных величин.

Правая часть неравенства убывает с ростом числа слагаемых n. Следствием неравенства (1.10.3) является закон больших чисел: *среднее арифметическое независимых случайных величин с одинаковыми математическими ожиданиями равными m и одинаковыми дисперсиями сходится по вероятности к математическому ожиданию m.*

Поясним термин «сходимость по вероятности». Говорят, что случайная последовательность $a_1, a_2,...$ сходится по вероятности к пределу a, если для любого $\varepsilon > 0$ имеет место равенство

$$\lim_{n\to\infty} P\left(\left|a_n - a\right| \geq \varepsilon\right) = 0.$$

В действительности неравенство (1.10.3) – очень слабая оценка вероятности, записанной в левой части. На самом деле эта вероятность убывает с ростом n экспоненциально (пропорционально e^{-Cn}, где $C > 0$), то есть гораздо быстрее, чем C/n. Удобство формулы (1.10.3) состоит в простоте ее доказательства и использования. Чтобы воспользоваться неравенством Чебышева достаточно знать только математическое ожидание и дисперсию элементов последовательности независимых с. в.

Еще одно замечание. Закон больших чисел справедлив не только для независимых случайных величин. Для произвольного стационарного случайного процесса введем в рассмотрение некоторую функцию f, вычисляемую по n последовательным значениям процесса. Последовательность значений функции f также образует случайный процесс. Если для любого n и для любой функции f для последовательности значений функции f справедлив закон больших чисел, процесс называют *эргодическим*.

Среднее арифметическое идущих подряд значений процесса $x_1,...,x_n$ представляет собой его *среднее по времени*. Математическое ожидание – это среднее по множеству значений, вычисленное по заданному распределению вероятностей. Это среднее называют средним по множеству реализаций. Поэтому часто эргодический процесс определяют как стационарный процесс, для которого усреднение по времени эквивалентно усреднению по множеству реализаций.

Таким образом, по определению, закон больших чисел справедлив для всех эргодических процессов. Примеры эргодических процессов – последовательность независимых одинаково распределенных случайных величин, простая неразложимая непериодическая цепь Маркова и т. п..

1.11. Прямая теорема кодирования для дискретного постоянного источника

По причинам, которые будут понятны позже, мы докажем теоремы кодирования только для дискретного постоянного источника, то есть источника без памяти.

Теорема 1.11.1. Пусть H – энтропия дискретного постоянного источника. Для любых $\varepsilon, \delta > 0$ существует n_0 такое, что для любого $n > n_0$ найдется равномерный код, кодирующий источник блоками длины n со скоростью $R \le H + \delta$ и вероятностью ошибки $P_e \le \varepsilon$.

Другими словами, при скорости кода хотя бы немного (на положительную величину δ) больше энтропии, увеличением длины кодируемых блоков можно при правильном кодировании добиться произвольно малой (меньше любого наперед заданного положительного ε) вероятности ошибки кодирования.

Доказательство.

Сначала мы укажем правило кодирования, затем выберем параметры кода так, чтобы скорость кода удовлетворяла условиям теоремы. После этого мы оценим вероятность ошибки и убедимся в том, что она может быть сделана сколь угодно малой.

Правило кодирования полностью описывается выбором множества $T \subseteq X^n$ однозначно кодируемых последовательностей сообщений источника. Выберем $T \subset X^n$ следующим образом

$$T = \left\{ \mathbf{x} : \left| \frac{1}{n} I(\mathbf{x}) - H \right| \le \delta_0 \right\}, \tag{1.11.1}$$

где через $I(\mathbf{x}) = -\log p(\mathbf{x})$ обозначена собственная информация последовательности $\mathbf{x} \in X^n$, а через δ_0 обозначена положительная константа, значение которой будет выбрано позже. Заметим, что множество T состоит из таких последовательностей, средняя собственная информация которых весьма близка к энтропии, если величина δ_0 достаточно мала.

Из (1.11.1) и определения собственной информации следует, что вероятности последовательностей множества T удовлетворяют неравенствам

$$2^{-n(H+\delta_0)} \le p(\mathbf{x}) \le 2^{-n(H-\delta_0)}. \tag{1.11.2}$$

Левая и правая части могут рассматриваться как оценки снизу и сверху для вероятностей последовательностей из T. Для оценки числа элементов в этом множестве заметим, что

$$1 \ge P(T) = \sum_{\mathbf{x} \in T} p(\mathbf{x}) \ge |\, T \,| \min_{\mathbf{x} \in T} p(\mathbf{x}) \ge |\, T \,| \, 2^{-n(H+\delta_0)}.$$

Следовательно,

$$|\, T \,| \le 2^{n(H+\delta_0)}. \tag{1.11.3}$$

По определению (1.9.1) скорость кода равна

$$R = \frac{\log |\, T \,|}{n} \le H + \delta_0. \tag{1.11.4}$$

Таким образом, при выборе T в соответствии с (1.11.1) скорость кода удовлетворяет условию теоремы при любом $\delta_0 \le \delta$.

Ошибка кодирования имеет место тогда, когда источник порождает последовательность, не входящую в множество T. Поэтому вероятность ошибки P_e удовлетворяет соотношениям

$$P_e = P(\mathbf{x} \notin T) = P\left(\left| \frac{1}{n} I(\mathbf{x}) - H \right| > \delta_0 \right) = P\left(\left| \frac{1}{n} \sum_{i=1}^{n} I(x_i) - H \right| > \delta_0 \right) \tag{1.11.5}$$

Последний переход учитывает то обстоятельство, что, по предположению теоремы, буквы источника независимы, и, в силу аддитивности собственной информации, в этом случае

информация последовательности равна сумме информации отдельных букв. Заметим теперь, что $\mathbf{M}\left[I(x)\right] = H$. Это означает, что к последнему выражению в (1.11.5) можно применить неравенство Чебышева. В результате имеем

$$P_e \le \frac{\mathbf{D}\left[I(x)\right]}{n\delta_o^2}. \qquad (1.11.6).$$

Из этого неравенства следует, что, поскольку $\mathbf{D}[I(x)]$ – ограниченная константа, то для любого источника и любой заданной наперед величины δ_0 можно теперь подобрать длину кодируемых блоков n так, чтобы вероятность ошибки была меньше любого заданного наперед ε. Положим теперь $\delta_0 = \delta/2$, $n_0 = 4\mathbf{D}[I(x)]/(\delta^2\varepsilon)$. Из (1.11.4) и (1.11.6) следует, что при $n \ge n_0$ справедливы неравенства $R < H + \delta$ и $P_e \le \varepsilon$, что и требовалось доказать.

■

В доказательстве теоремы участвовало множество T, определенное соотношением (1.11.1). Его называют множеством типичных последовательностей. Это множество играет важную роль в теории информации. Мы уже установили некоторые его свойства, некоторые другие свойства будут установлены при доказательстве обратной теоремы. Затем мы суммируем все эти свойства в специально посвященном этому множеству параграфе.

1.12. Обратная теорема кодирования для дискретного постоянного источника

Мы приступаем к формулировке и доказательству обратной теоремы кодирования. Нам предстоит доказать, что невозможно обеспечить кодирование с малой ошибкой при скорости меньшей, чем энтропия источника. Иными словами, сколь угодно изобретательный инженер не сможет сжать информацию равномерными кодами сильнее, чем это гарантируется приведенной выше прямой теоремой кодирования. Возможно, именно такого рода утверждения послужили причиной того, что в свое время теория информации считалась идеалогически «неправильной» наукой. Итак,

Теорема 1.12.1. Для дискретного постоянного источника с энтропией H существуют $\varepsilon > 0$ такое, что для любого $\delta > 0$ для любого равномерного кода со скоростью $R \le H - \delta$ вероятность ошибки удовлетворяет неравенству $P_e \ge \varepsilon$.

Иными словами, если скорость хотя бы ненамного меньше энтропии, то вероятность ошибки уже не может быть сколь угодно малой. На самом деле, мы убедимся в том, что при больших длинах блоков n вероятность ошибки будет довольно большой.

Доказательство.

Представим себе оппонента, который не согласен с утверждением теоремы и считает, что сколь угодно точное кодирование возможно при скорости меньшей, чем энтропия источника. Чтобы переубедить его, достаточно указать для заданного источника некоторую строго положительную независящую от длины кодов величину ε, которая была бы оценкой снизу вероятности ошибки любого кода со скоростью $R \le H - \delta$.

Код полностью описывается выбором множества T_1 однозначно кодируемых последовательностей. По определению, скорость кода равна $R = \log|T_1|/n$, следовательно, код содержит

$$|T_1| = 2^{nR} \le 2^{n(H-\delta)} \qquad (1.12.1)$$

кодовых слов.

Вместо вычисления нижней оценки для вероятности ошибки P_e построим оценку сверху для вероятности правильного кодирования

$$P_c = 1 - P_e = \sum_{\mathbf{x}\in T_1} p(\mathbf{x}). \qquad (1.12.2)$$

Введем вспомогательное множество

$$T = \left\{ \mathbf{x} : \left| \frac{1}{n} I(\mathbf{x}) - H \right| \le \delta_0 \right\}, \tag{1.12.3}$$

где через $I(\mathbf{x}) = -\log p(\mathbf{x})$ обозначена собственная информация последовательности $\mathbf{x} \in X^n$, а через δ_0 – положительная константа, значение которой будет выбрано позже. (Хотя это и не важно для доказательства, отметим, что (1.12.3) совпадает с (1.11.1), т.е. T совпадает с множеством однозначно кодируемых последовательностей, использованным для доказательства прямой теоремы кодирования).

Разобьем теперь сумму в правой части (1.12.2) на две суммы

$$P_c = \sum_{\mathbf{x} \in T_1 \cap T} p(\mathbf{x}) + \sum_{\mathbf{x} \in T_1 \cap T^c} p(\mathbf{x}), \tag{1.12.4}$$

где через T^c обозначено дополнение к множеству T. Оценим вторую сумму следующим образом:

$$\sum_{\mathbf{x} \in T_1 \cap T^c} p(\mathbf{x}) \le \sum_{\mathbf{x} \in T^c} p(\mathbf{x}) = P(T^c) = P(\mathbf{x} \notin T).$$

Эту вероятность мы уже оценивали при доказательстве прямой теоремы. Точно так же, как при переходе от (1.11.5) к (1.11.6), с помощью неравенства Чебышева получаем оценку

$$\sum_{\mathbf{x} \in T_1 \cap T^c} p(\mathbf{x}) \le \frac{\mathbf{D}[I(x)]}{n\delta_o^2}. \tag{1.12.5}$$

Для первой из двух сумм в (1.12.4) имеем

$$\sum_{\mathbf{x} \in T_1 \cap T} p(\mathbf{x}) \le | T_1 \cap T | \max_{\mathbf{x} \in T_1 \cap T} p(\mathbf{x}) \le | T_1 | \max_{\mathbf{x} \in T_1 \cap T} p(\mathbf{x}) \le | T_1 | \max_{\mathbf{x} \in T} p(\mathbf{x}). \tag{1.12.6}$$

Здесь мы сначала оценили сумму сверху, умножив число слагаемых на максимальное слагаемое. Затем использовали неравенство $|T_1 \cap T| \le |T_1|$. Последний переход основан на том, что, при расширении области поиска максимума максимальное значение либо увеличится, либо останется прежним.

Подставим в (1.12.6) оценку мощности кода (1.12.1), и оценку максимальной вероятности последовательностей из T (1.11.2). Получим

$$\sum_{\mathbf{x} \in T_1 \cap T} p(\mathbf{x}) \le 2^{n(H-\delta)} 2^{-n(H-\delta_0)} = 2^{-n(\delta-\delta_0)}. \tag{1.12.7}$$

Подстановка (1.12.5) и (1.12.7) в (1.12.4) приводит к следующей оценке сверху для вероятности правильного кодирования:

$$P_c \le 2^{-n(\delta-\delta_0)} + \frac{\mathbf{D}[I(x)]}{n\delta_o^2}. \tag{1.12.8}$$

Пусть δ - произвольное положительное число. Выберем теперь значение параметра δ_0 равным $\delta_0 = \delta/2$. При таком выборе правая часть неравенства (1.12.8) убывает с увеличением n. Обозначим через n_0 такое значение длины последовательностей n, что при $n \ge n_0$ справедливо неравенство $P_c \le 1/2$. При этом, конечно, вероятность ошибки $P_e \ge 1/2$. Теперь обозначим через ε_0 минимальную вероятность ошибочного кодирования по всем $n = 1,2,...,n_0 - 1$. Примем $\varepsilon = \min\{\varepsilon_0, 1/2\}$. Получаем неравенство $P_e \ge \varepsilon$, справедливое при всех $n = 1,2,...$.

Таким образом, мы нашли независящую от длины кода n нижнюю границу вероятности ошибки ε для всех кодов со скоростью $R \le H - \delta$. Тем самым теорема доказана.

Как и в доказательстве прямой теоремы, мы воспользовались свойствами множества типичных последовательностей T. Эвристическое обоснование теорем

кодирования и более подробное обсуждение множества T приведено в следующем параграфе.

1.13. Множество типичных последовательностей для дискретного постоянного источника

Рассмотрим дискретный постоянный источник, порождающий последовательность независимых сообщений из ансамбля $X = \{x, p(x)\}$. Множество типичных последовательностей длины n обозначим как $T_n(\delta)$. Оно определено выше как множество последовательностей, средняя собственная информация которых близка к энтропии $H(X)$, то есть

$$T_n(\delta) = \left\{ \mathbf{x} : \left| \frac{1}{n} I(\mathbf{x}) - H(X) \right| \le \delta \right\},$$

где δ – некоторая положительная константа. Подытожим в виде теоремы некоторые свойства множества $T_n(\delta)$, в основном, уже доказанные и использованные при доказательстве теорем кодирования для дискретного постоянного источника.

Теорема 1.13.1. Для любого $\delta > 0$ имеют место следующие утверждения:

1. $\lim\limits_{n\to\infty} P\left(T_n(\delta)\right) = 1$.
2. Для любого натурального n справедливо неравенство $\left|T_n(\delta)\right| \le 2^{n(H(X)+\delta)}$.
3. Для любого $\varepsilon > 0$ существует n_0 такое, что
$$\left|T_n(\delta)\right| \ge (1-\varepsilon)2^{n(H(X)-\delta)}$$
при всех $n \ge n_0$.
4. $2^{-n(H(X)+\delta)} \le p(\mathbf{x}) \le 2^{-n(H(X)-\delta)}$ для любого $\mathbf{x} \in T_n(\delta)$.

Другими словами, теорема в пункте 1 утверждает, что асимптотически почти все порожденные источником последовательности типичны, т.е. принадлежат $T_n(\delta)$. Второе и третье свойства указывают на то, что при достаточно больших n в $T_n(\delta)$ содержится приблизительно $2^{nH(X)}$ последовательностей. Тем самым установлена простая связь между энтропией ансамбля и мощностью множества типичных последовательностей сообщений ансамбля. Наконец, из четвертого свойства мы узнаем, что все последовательности, входящие в $T_n(\delta)$ асимптотически равновероятны, они имеют приблизительно равные вероятности примерно равные $2^{-nH(X)}$.

Доказательство. Первое утверждение следует из (1.11.5) и (1.11.6). Второе совпадает с (1.11.3), четвертое – с (1.11.2). Остановимся на третьем утверждении.

Из первого утверждения следует, что для любого $\varepsilon > 0$ найдется n_0 такое, что при $n > n_0$ имеет место неравенство

$$P\left(T_n(\delta)\right) \ge 1 - \varepsilon. \qquad (1.13.1)$$

Оценивая вероятность $T_n(\delta)$ как произведение числа элементов в множестве на величину максимального элемента и применяя утверждение 4, получаем

$$P\left(T_n(\delta)\right) \le \left|T_n(\delta)\right| \max_{\mathbf{x}\in T_n(\delta)} p(\mathbf{x}) \le \left|T_n(\delta)\right| 2^{-n(H(X)-\delta)}. \qquad (1.13.2)$$

Из (1.13.1) и (1.13.2) следует утверждение 3. ■

Теперь понятно, почему справедлива прямая теорема кодирования. Из свойства 1 следует, что для получения малой вероятности ошибки достаточно кодировать последовательности из $T_n(\delta)$. Поскольку таких последовательностей примерно $2^{nH(X)}$,

для передачи номера последовательности достаточно затратить $nH(X)$ бит или $H(X)$ бит на одно сообщение.

Также просто объясняется справедливость обратной теоремы. Если взаимно однозначно кодируется M последовательностей, то суммарная вероятность однозначно кодируемых последовательностей из $T_n(\delta)$ имеет порядок $M2^{-nH(X)}$. Вероятность ошибки примерно равна $1 - M2^{-nH(X)}$. Пока число M меньше $2^{nH(X)}$ (соответственно, скорость кода меньше $H(X)$), произведение $M2^{-nH(X)}$ будет маленьким, а вероятность ошибки будет заведомо большой.

Посмотрим теперь подробнее, какие именно последовательности являются типичными. Обозначим через $\tau_x(\mathbf{x})$ число появлений буквы x в последовательности $\mathbf{x}$. Набор чисел $\boldsymbol{\tau}(\mathbf{x}) = \{\tau_x(\mathbf{x}), x \in X\}$ называют композицией последовательности $\mathbf{x}$.

Для дискретного постоянного источника вероятность последовательности $\mathbf{x} = (x_1, ..., x_n)$ может быть записана в виде

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i) = \prod_{x \in X} p(x)^{\tau_x(\mathbf{x})} .$$

Это представление получено изменением порядка сомножителей в произведении. Собственная информация последовательностей в расчете на букву равна

$$\frac{1}{n} I(\mathbf{x}) = -\sum_{x \in X} \frac{\tau_x(\mathbf{x})}{n} \log p(x) .$$

Эта величина близка к энтропии $H(X)$, если

$$\frac{\tau_x(\mathbf{x})}{n} \approx p(x) .$$

Мы пришли к естественному результату: множество типичных последовательностей состоит из тех последовательностей, в которых доля элементов x приблизительно равна вероятности символа x.

Раздел 2.

Неравномерное кодирование дискретных источников.

В предыдущем разделе мы убедились в том, что источники информации, вообще говоря, избыточны в том смысле, что, при эффективном кодировании можно уменьшить затраты на передачу или хранение порождаемой ими информации. Для представления данных потребуется тем меньше бит, чем меньше энтропия. Это значит, что для эффективного кодирования удобны источники, в которых некоторые буквы (последовательности букв) имеют существенно большую вероятность, чем другие буквы (последовательности). Не нужно долго изучать теорию информации, чтобы догадаться, что для таких источников нужно использовать кодирование, сопоставляющее часто встречающимся сообщениям короткие кодовые комбинации, а редким сообщениям – длинные. Реализация этой простой идеи является предметом исследования в данном разделе.

Мы начнем с кодирования отдельных букв источника. Оптимальным побуквенным кодом является известный код Хаффмана. Однако, для того, чтобы понять, как правильно кодировать последовательности, нам придется изучить неоптимальные побуквенные коды – код Шеннона и код Гилберта-Мура. От них – один шаг к пониманию арифметического кодирования, которое позволяет предельно эффективно кодировать длинные последовательности и обладает при этом относительно невысокой сложностью.

2.1. Постановка задачи неравномерного побуквенного кодирования

Предположим, что для некоторого дискретного источника X с известным распределением вероятностей $\{p(x), x \in X\}$ требуется построить эффективный неравномерный двоичный код над алфавитом $A = \{a\}$. Как и в предыдущем разделе, мы сосредоточим внимание на двоичных кодах, т.е. мы предполагаем, что $A = \{0,1\}$. Дело в том, что, во-первых, все идеи в полной мере иллюстрируются на этом примере. Во-вторых, обобщение на случай произвольного алфавита не представляет никакой трудности. Помимо этого на практике для кодирования источников используются почти исключительно двоичные коды.

В качестве примера источника рассмотрим алфавит русского языка. Сразу же вспоминается азбука Морзе, которая сопоставляет каждой букве комбинацию точек «•» и тире «–». Например, часто встречающейся букве «е» соответствует комбинация «•», а более редкой букве «ч» соответствует комбинация «– – – •». Однако, более пристальный взгляд убеждает, что мы не получим хорошего кода просто заменив точки нулями, а тире – единицами. Нам будет не хватать пауз, разделяющих символы внутри букв (эта пауза соответствует интервалу времени равному времени передачи точки), пауз, разделяющих буквы (3 точки), пауз, разделяющих слова (7 точек). По сути дела код Морзе – это недвоичный код.

Нам необходим такой двоичный код, который допускает однозначное разделение последовательности кодовых слов на отдельные кодовые слова без использования каких-либо дополнительных символов. Это требование называют свойством *однозначной декодируемости.*

Неравномерный побуквенный код $C = \{\mathbf{c}\}$ объема $|C| = M$ над алфавитом A определяется как произвольное множество последовательностей одинаковой или различной длины из букв алфавита A. Код является *однозначно декодируемым*, если любая последовательность символов из A единственным способом разбивается на

отдельные кодовые слова.

Пример 2.1.1. Для источника $X = \{0,1,2,3\}$ среди четырех кодов

a) $C_1 = \{00,01,10,11\}$ **;**

b) $C_2 = \{1,01,001,000\}$ **;**

c) $C_3 = \{1,10,100,000\}$ **;**

d) $C_4 = \{0,1,10,01\}$ **;**

первые три кода однозначно декодируемы, последний код – нет.

Первый код этого примера – равномерный код. Понятно, что любой равномерный код может быть однозначно декодирован.

Для декодирования второго кода можно применить следующую стратегию. Декодер считывает символ за символом, и каждый раз проверяет, не совпадает ли полученная последовательность с одним из кодовых слов. В случае успеха соответствующее сообщение выдается получателю, и декодер приступает к декодированию следующего сообщения. В случае кода C_2 неоднозначности не может быть, поскольку ни одно слово не является продолжением другого.

Если ни одно кодовое слово не является началом другого, код называется *префиксным*. Префиксные коды являются однозначно декодируемыми.

Код C_3 заведомо не префиксный. Тем не менее, мы утверждаем, что он – однозначно декодируемый. Каждое слово кода C_3 получено переписыванием в обратном порядке соответствующего слова кода C_2. Для декодирования последовательности кодовых слов кода C_3 можно переписать принятую последовательность в обратном порядке и для декодирования использовать декодер кода C_2. Таким образом, мы приходим к следующему выводу.

Префиксность – достаточное, но не необходимое условие однозначной декодируемости.

Графически удобно представлять префиксные коды в виде кодовых деревьев. В частности, кодовое дерево кода C_2 примера 2.1. представлено на рис.2.1.1.

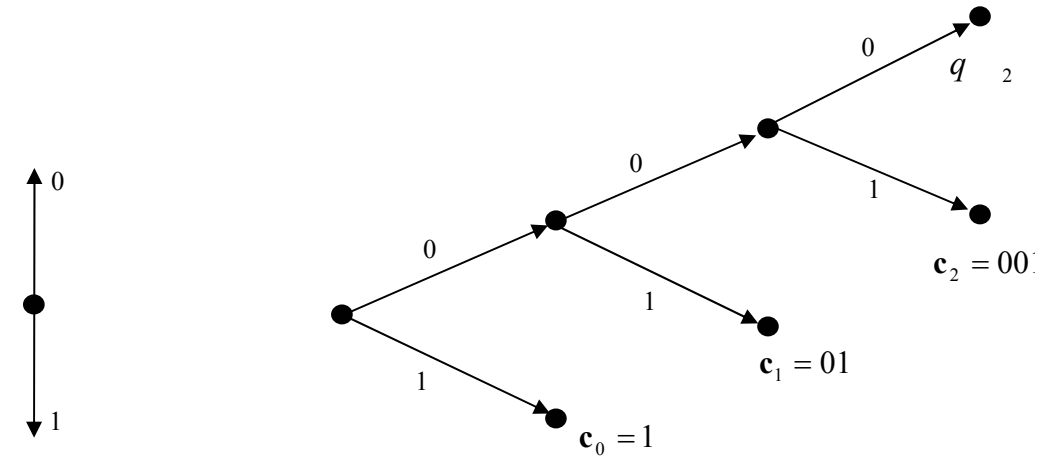


Рис.2.1.1. Кодовое дерево кода C_2 примера 2.1.

Узлы дерева размещаются на ярусах. На начальном (нулевом) ярусе расположен один узел, называемый *корнем дерева.* Узлы следующих ярусов связаны с узлами предыдущих ярусов ребрами. В случае двоичного кода из каждого узла исходит не более двух ребер. Ребрам приписаны кодовые символы. В данном примере принято соглашение

о том, что ребру, ведущему вверх, приписывается символ 0, а ребру, ведущему вниз – символ 1. Таким образом, каждой вершине дерева соответствует последовательность, считываемая вдоль пути, связывающего данный узел с корнем дерева.

Узел называется *концевым*, если из него не исходит ни одного ребра. Код называют *древовидным*, если в качестве кодовых слов он содержит только кодовые слова, соответствующие концевым вершинам кодового дерева.

Древовидность кода и префиксность – синонимы в том смысле, что всякий древовидный код является префиксным, и всякий префиксный код может быть представлен с помощью кодового дерева. Мы будем рассматривать только префиксные коды. В связи с этим возникает вопрос: не потеряли ли мы оптимальное решение задачи, сузив класс кодов, среди которых мы ищем наилучшие? Ниже мы убедимся в том, что ответ на этот вопрос отрицательный.

Обсудим теперь, какие именно префиксные коды считать хорошими. Поскольку сама цель рассмотрения неравномерного кодирования состояла в уменьшении затрат на передачу сообщений, логично выбрать в качестве критерия качества кода среднюю длину кодовых слов. Рассмотрим источник $X = \{1,...,M\}$, который порождает буквы с вероятностями $\{p_1,..., p_M\}$. Предположим, что для кодирования букв источника выбран код $C = \{\mathbf{c}_1,...,\mathbf{c}_M\}$ с длинами кодовых слов $\text{length}(\mathbf{c}_1) = l_1,...,\text{length}(\mathbf{c}_M) = L_M$ соответственно. *Средней длиной кодовых слов* называется величина

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

Пример 2.1.2. Рассмотрим источник и коды из примера 2.1.1. Подсчитайте среднюю длину кодовых слов кодов C_1 и C_2 для двух распределений вероятностей на буквах источника:

a) $p_0 = p_1 = p_2 = p_3 = 1/4$;

b) $p_0 = 1/2$; $p_1 = 1/4$; $p_3 = p_4 = 1/8$;

c) $p_0 = p_1 = 1/8$; $p_2 = 1/4$; $p_3 = 1/2$.

Результаты расчетов показывают, что не всегда и не любой неравномерный код эффективен.

Еще один немаловажный аспект, который должен быть учтен при сравнении способов неравномерного кодирования, – это сложность реализации кодирования и декодирования. В зависимости от области применения может изменяться и значение допустимой сложности, и само понятие сложности. Например, при реализации алгоритма на универсальном компьютере практически отсутствует ограничение на размер оперативной памяти, но имеет значение вычислительная сложность. При реализации кодера и декодера в виде интегральной микросхемы или с помощью специализированного сигнального процессора, напротив, память является определяющим параметром.

Как мы увидим далее, побуквенные коды часто используются как составная часть более сложных алгоритмов. В этих случаях играет роль не только сложность собственно кодирования и декодирования буквы, но и сложность построения или модификации кода при изменении статистических данных об источнике.

Подводя итог, мы формулируем задачу побуквенного неравномерного кодирования как задачу построения однозначно декодируемого кода с наименьшей средней длиной кодовых слов при заданных ограничениях на сложность.

2.2. Неравенство Крафта

Вернемся к примеру 2.1.1 и сравним коды C_1 и C_2 объема 4. Код C_1 равномерный, все слова имеют одинаковую длину 2. Нельзя ли выбрать слова короче? Действительно, в

коде C_2 есть одно слово длины 1, но чтобы сохранить объем кода и префиксность, пришлось на единицу увеличить длины двух других кодовых слов. Этот пример показывает, что требование префиксности накладывает жесткие ограничения на множество длин кодовых слов и не дает возможности выбирать кодовые слова слишком короткими. Формально эти ограничения записываются в виде изящного неравенства, называемого неравенством Крафта.

Теорема 2.2.1. Необходимым и достаточным условием существования префиксного кода объема M с длинами кодовых слов $l_1,...,l_M$ является выполнение неравенства

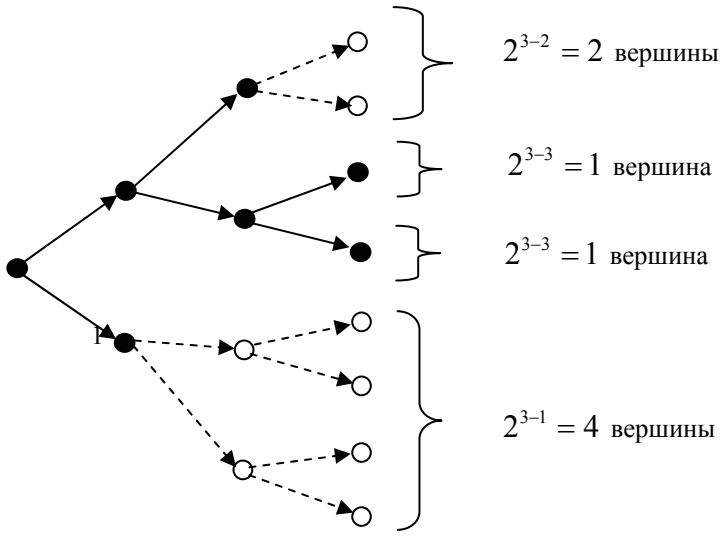
$$\sum_{i=1}^{M} 2^{-l_i} \le 1. \qquad (2.2.1)$$

Доказательство. Начнем *с необходимости*. Мы должны убедиться в том, что неравенство (2.2.1) верно для любого префиксного кода.

Рассмотрим двоичное кодовое дерево произвольного префиксного кода объема M с длинами кодовых слов $l_1,...,l_M$. Выберем целое число L, такое, что $L \ge \max_i l_i$. Продолжим все пути в дереве до яруса с номером L. На последнем ярусе мы получим 2^L вершин. Заметим, что концевая вершина исходного дерева, расположенная на глубине l_i, имеет 2 потомка на глубине $l_i + 1$, 4 потомка на глубине $l_i + 2$, и т.д. На глубине L будет 2^{L-l_i} потомков этой вершины. Множества потомков различных концевых вершин не пересекаются, поэтому суммарное число потомков не превышает общего числа вершин на ярусе L. Получаем неравенство

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

Поделив обе части на 2^L, получим требуемый результат. Рис.2.2.1 иллюстрирует доказательство необходимости выполнения неравенства Крафта на примере кода из 4 слов. В этом примере $l_1 = 2$, $l_2 = l_3 = 3$, $l_4 = 1$, $L = 4$. Кодовое дерево показано сплошными линиями а ребра, появившиеся при продлении дерева до яруса с номером L показаны пунктиром.



Всего вершин $2+1+1+4=8$; $\Rightarrow$ $1/4+1/8+1/8+1/2=1$

Рис. 2.2.1. Пояснение к доказательству справедливости неравенства Крафта для префиксного кода

Чтобы убедиться в *достаточности*, нужно показать, что из справедливости (2.2.1) вытекает существование кода с заданным набором длин кодовых слов. Построим такой код. Без потери общности можем считать числа l_i упорядоченными по возрастанию.

Из общего числа 2^{l_1} вершин на ярусе l_1 выберем одну любую, сделаем ее концевой и закрепим ее за первым кодовым словом. Продолжим оставшиеся вершины до яруса l_2. Из общего числа возможных вершин нужно исключить $2^{l_2-l_1}$ вершин, которые принадлежат поддереву, начинающемуся в узле, соответствующем первому слову. На ярусе l_2 останется

$$2^{l_2} - 2^{l_2-l_1} \geq 1$$

вершин. Последнее неравенство следует из (2.2.1), в чем нетрудно убедиться, поделив его правую и левую часть на 2^{l_2}. Сделаем одну из них концевой и закрепим ее за вторым словом. Аналогично, для третьего слова мы получим множество из

$$2^{l_3} - 2^{l_3-l_2} - 2^{l_3-l_1} \geq 1$$

вершин. В силу (2.2.1) всегда найдется одна для третьего слова. Продолжая построение, на последнем ярусе с номером l_M мы получим

$$2^{l_M} - 2^{l_M-l_{M-1}} - 2^{l_M-l_{M-2}} - \ldots - 2^{l_M-l_1}$$

вершин. Простые выкладки показывают, что это число не меньше 1, если неравенство (2.2.1) верно. Выбрав эту вершину для последнего слова, мы завершим построение префиксного кода. Рис. 2.2.2 иллюстрирует процесс построения кодового дерева для набора длин кодовых слов $l_1 = 1$, $l_2 = 2$, $l_3 = l_4 = 3$. ■

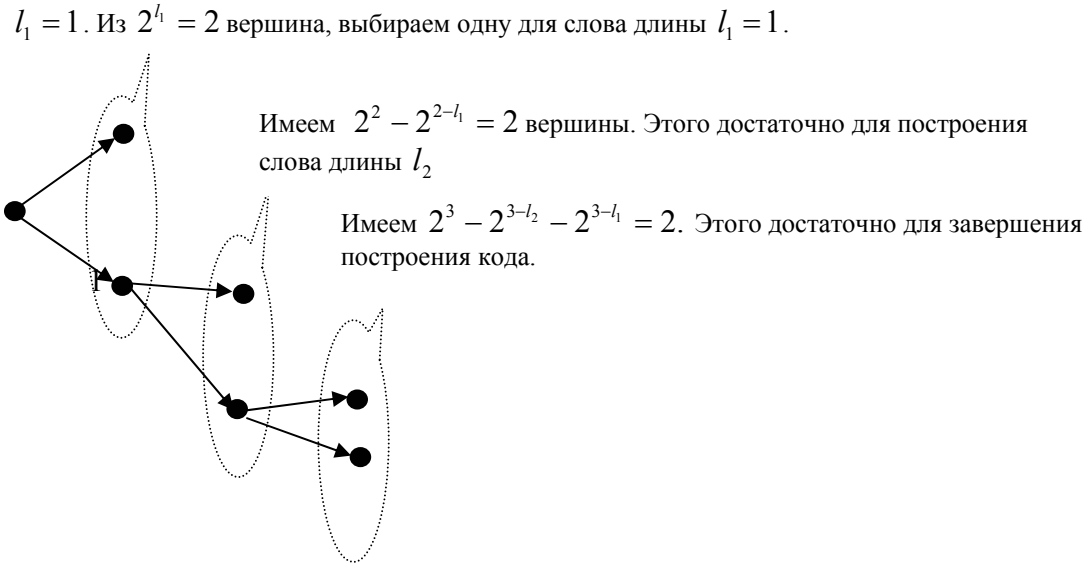


Рис. 2.2.2. Построение префиксного кода с длинами слов, удовлетворяющими неравенству Крафта

В рассматриваемом примере неравенство Крафта выполняется с равенством. Внимательно просмотрев доказательство необходимости в Теореме 2.2.1, убеждаемся, что для достижения равенства в (2.2.1) кодовое дерево должно быть полным, т.е. каждая промежуточная вершина дерева должна иметь ровно 2 потомка и всем концевым вершинам должны быть сопоставлены кодовые слова.

Неравенство Крафта как бы ограничивает снизу длины кодовых слов префиксного кода заданного объема M. На этом будет впоследствии построено доказательство

обратной теоремы кодирования. В связи с этим важно быть уверенным, что оно имеет место не только для древовидных (префиксных) кодов, но и для любых других однозначно декодируемых кодов. Это утверждение является содержанием следующей теоремы.

Теорема 2.2.2. Для любого однозначно декодируемого двоичного кода объема M с длинами кодовых слов $l_1,...,l_M$ справедливо неравенство

$$\sum_{i=1}^{M} 2^{-l_i} \le 1. \qquad (2.2.2)$$

Доказательство. Без потери общности можно считать l_M наибольшей из длин кодовых слов. Выберем некоторое целое положительное число N и запишем N-ю степень левой части неравенства в виде

$$\left(\sum_{i=1}^{M} 2^{-l_i}\right)^N = \underbrace{\left(\sum_{i_1=1}^{M} 2^{-l_{i_1}}\right)\cdots\left(\sum_{i_N=1}^{M} 2^{-l_{i_N}}\right)}_{N \text{ сомножителей}} = \sum_{i_1=1}^{M}\cdots\sum_{i_N=1}^{M} 2^{-(l_{i_1}+\ldots+l_{i_N})}.$$

В правой части имеем M^N слагаемых. Их мы перегруппируем, упорядочив по величине суммы длин кодовых слов $l_{i_1}+\ldots+l_{i_N}$. Обозначим через A_L число таких последовательностей кодовых слов, сумма длин которых равна $l_{i_1}+\ldots+l_{i_N}=L.$ Тогда

$$\left(\sum_{i=1}^{M} 2^{-l_i}\right)^N = \sum_{L=1}^{Nl_M} A_L 2^{-L}.$$

Теперь заметим, что A_L, т.е. число таких последовательностей из N кодовых слов, суммарная длина которых в точности равна L не может быть больше общего числа двоичных последовательностей длины L. Таким образом, $A_L \le 2^L.$ Поэтому

$$\left(\sum_{i=1}^{M} 2^{-l_i}\right)^N \le \sum_{L=1}^{Nl_M} 2^L 2^{-L} = Nl_M.$$

Извлечем из обеих частей корень N-й степени. Получим

$$\sum_{i=1}^{M} 2^{-l_i} \le \left(Nl_M\right)^{1/N}..$$

Неравенство справедливо при любых N. Перейдя к пределу при $N \to \infty$ в правой части получим 1, что и доказывает теорему. ■

2.3. Прямая теорема побуквенного неравномерного кодирования

Начнем непосредственно с формулировки теоремы кодирования.

Теорема 2.3.1. Для ансамбля $X = \{x, p(x)\}$ с энтропией H существует побуквенный неравномерный префиксный код со средней длиной кодовых слов $\bar{l} \le H + 1$**.**

Доказательство. У нас уже есть опыт доказательства прямой теоремы кодирования для случая равномерного кодирования источников. Доказательство той теоремы было конструктивным в том смысле, что в явной форме был указан способ построения кода, удовлетворяющего условиям теоремы. Здесь мы могли бы поступить также. Однако мы воспользуемся случаем, чтобы продемонстрировать пример неконструктивного доказательства. Опираясь на неравенство Крафта, мы установим существование кода, не указав, как можно его построить. Этот подход достаточно типичен для теории информации. Больше того, для многих теорем кодирования, доказанных несколько десятилетий назад, конструктивных методов доказательства не известно до сих пор.

Итак, рассмотрим источник над алфавитом $X = \{1,...,M\}$ с вероятностями букв соответственно $p_1,..., p_M$. Сопоставим букве x_m кодовое слово длины $l_m = \lceil -\log p_m \rceil$ при

$m = 1,...,M$. (Здесь и ниже запись $\lfloor a \rfloor$ означают округление числа a вниз до ближайшего целого, а запись $\lceil a \rceil$ - округление числа a вверх до ближайшего целого). Префиксный код с таким набором длин кодовых слов в соответствии с Теоремой 2.2.1 существует, поскольку длины кодовых слов удовлетворяют неравенству Крафта

$$\sum_{m=1}^{M} 2^{-l_m} = \sum_{m=1}^{M} 2^{-\lceil -\log p_m \rceil} \le \sum_{m=1}^{M} 2^{\log p_m} = \sum_{m=1}^{M} p_m = 1.$$

Средняя длина кодовых слов кода

$$\bar{l} = \sum_{m=1}^{M} p_m l_m = \sum_{m=1}^{M} p_m \lceil -\log p_m \rceil \le \sum_{m=1}^{M} p_m (-\log p_m + 1) \le H + \sum_{m=1}^{M} p_m = H + 1,$$

что и требовалось доказать. ■

Обсудим полученную оценку длины кодовых слов. Мы знаем (ниже это будет установлено точно с помощью обратной теоремы кодирования), что достижимая скорость кодирования примерно равна энтропии. Теорема гарантирует, что средняя длина слов хорошего кода отличается от энтропии не больше, чем на 1. Если энтропия велика, то проигрыш по сравнению с минимально достижимой скоростью можно считать небольшим. Но предположим, что $H < 1$. Например, $H = 0{,}1$. Теорема гарантирует, что существует код со средней длиной кодовых слов не больше 1,1 бита. Но нам хотелось бы затрачивать на передачу одного сообщение примерно в 10 раз меньше бит! Этот пример показывает, что либо теорема дает неточную оценку, либо побуквенное кодирование в этом случае не эффективно.

На этом же примере мы убедимся в том, что теорема достаточно точна и ее результат не может быть улучшен, если не использовать никакой дополнительной информации об источнике. Действительно, предположим, что дан двоичный источник $X = \{0,1\}$ с вероятностями букв $\{\varepsilon, 1-\varepsilon\}$. Минимально достижимая длина кодовых слов наилучшего кода, очевидно, равна 1. Теорема говорит, что средняя длина кодовых слов не больше $1 + h(\varepsilon)$, т.е. стремится к 1 при $\varepsilon \to 0$. Таким образом, для данного примера двоичного источника теорема точна.

2.4. Обратная теорема неравномерного побуквенного кодирования

Обратная теорема неравномерного кодирования устанавливает нижнюю границу средней длины кодовых слов любого однозначно декодируемого кода.

Теорема 2.4.1. Для любого однозначно декодируемого кода дискретного источника $X = \{x, p(x)\}$ с энтропией H средняя длина кодовых слов $\bar{l}$ удовлетворяет неравенству

$$\bar{l} \ge H. \qquad (2.4.1)$$

Другими словами, не существует кода со средней длиной кодовых слов меньше H и обладающего свойством однозначной декодируемости.

Доказательство. Пусть $l(x)$ обозначает длину кодового слова для сообщения x. Имеем

$$H - \bar{l} = -\sum_{x \in X} p(x) \log p(x) - \sum_{x \in X} p(x) l(x) = \sum_{x \in X} p(x) \log \frac{2^{-l(x)}}{p(x)}.$$

Применяя уже знакомое нам неравенство для логарифма

$$\log x \le (x-1) \log e,$$

получим

$$H - \bar{l} \le \log e \sum_{x \in X} p(x) \left(\frac{2^{-l(x)}}{p(x)} - 1 \right) = \log e \left(\sum_{x \in X} 2^{-l(x)} - \sum_{x \in X} p(x) \right) \le \log e \left(1 - \sum_{x \in X} p(x) \right) = 0. \qquad (2.4.2)$$

Здесь мы использовали неравенство Крафта и условие нормировки вероятностей. Из (2.4.2) следует утверждение теоремы. ■

Обсудим вопрос о том, при каких условиях возможно равенство в обратной теореме. Для этого равенство должно иметь место в первом и втором неравенствах в (2.4.2). Для этого при каждом x должно выполняться соотношение $p(x) = 2^{-l(x)}$. В то же время для такого распределения вероятностей существует полный префиксный код с длинами кодовых слов $l(x) = \lceil -\log p(x) \rceil = -\log p(x)$. Для этого кода неравенство Крафта выполняется с равенством средняя длина кодового слова $\bar{l} = H$.

Таким образом мы установили справедливость следующего утверждения.

Следствие. Необходимым условием существования кода со средней длиной кодовых слов $\bar{l} = H$ необходимо, чтобы все вероятности сообщений $x \in X$ имели вид $p(x) = 2^{-l(x)}$, где $\{l(x)\}$ – целые положительные числа.

2.4. Оптимальный побуквенный код – код Хаффмена

Предложенный Хаффменом алгоритм построения оптимальных неравномерных кодов – одно из самых важных достижений теории информации как с теоретической, так и с прикладной точек зрения. Трудно поверить, но этот алгоритм был придуман в 1952 г. студентом Дэвидом Хаффменом в процессе выполнения домашнего задания.

Рассмотрим ансамбль сообщений $X = \{1,...,M\}$ с вероятностями сообщений $\{p_1,...,p_M\}$. Без потери общности мы считаем сообщения упорядоченными по убыванию вероятностей, т.е. $p_1 \le p_2 \le ... \le p_M$. Наша задача состоит в построении оптимального кода, т.е. кода с наименьшей возможной средней длиной кодовых слов. Понятно, что при заданных вероятностях такой код может не быть единственным, возможно существование семейства оптимальных кодов. Мы установим некоторые свойства всех кодов этого семейства. Эти свойства подскажут нам простой путь к нахождению одного из оптимальных кодов.

Пусть двоичный код $C = \{\mathbf{c}_1,...,\mathbf{c}_M\}$ с длинами кодовых слов $\{l_1,...,l_M\}$ оптимален для рассматриваемого ансамбля сообщений.

Свойство 1. Если $p_i < p_j$, то $l_i \ge l_j$.

Доказательство. Свойство легко доказывается методом «от противного». Предположим, что $l_i < l_j$. Рассмотрим другой код C', в котором сообщению x_i соответствует слово $\mathbf{c}_j$, а сообщению x_j – слово $\mathbf{c}_i$. Нетрудно убедиться в том, что средняя длина кодовых слов для кода C' меньше, чем для кода C, что противоречит предположению об оптимальности кода C.

Свойство 2. Не менее двух кодовых слов имеют одинаковую длину $l_M = \max_m l_m$.

Доказательство. Если предположить, что имеется только одно слово максимальной длины, то соответствующее кодовое дерево будет неполным. Очевидно, слово максимальной длины можно будет сделать короче по меньшей мере на 1 символ. При этом уменьшится средняя длина кодовых слов, что противоречит предположению об оптимальности кода.

Свойство 3. Среди кодовых слов длины $l_M = \max_m l_m$ найдутся 2 слова, отличающиеся только в одном последнем символе.

Доказательство. Согласно предыдущему свойству, 2 слова длины l_M существуют в любом оптимальном коде. Рассмотрим концевой узел, соответствующий одному из слов максимальной длины. Чтобы дерево было полным, должен существовать узел, имеющий общий предшествующий узел с данным узлом. Соответствующие двум концевым вершинам кодовые слова имеют одинаковую длину l_M и отличаются в одном последнем символе.

Прежде, чем сформулировать следующее свойство, введем дополнительные

обозначения. Для рассматриваемого ансамбля $X = \{1,...,M\}$ и некоторого кода C, удовлетворяющего свойствам 1-3 введем вспомогательный ансамбль $X' = \{1,...,M-1\}$, сообщениям которого сопоставим вероятности $\{p'_1,...,p'_{M-1}\}$ следующим образом

$$p'_1 = p_1,\ \ldots,\ p'_{M-2} = p_{M-2},\ p'_{M-1} = p_{M-1} + p_M .$$

Из кода C построим код C' для ансамбля X', приписав сообщениям $x'_1,...x'_{M-2}$ те же кодовые слова, что и в коде C, т.е. $\mathbf{c}'_i = \mathbf{c}_i,\ i = 1,...,M-2$, а сообщению x'_{M-1} – слово $\mathbf{c}'_{M-1}$, представляющее собой общую часть слов $\mathbf{c}_{M-1}$ и $\mathbf{c}_M$ (согласно свойству 3 эти два кодовых слова отличаются только в одном последнем символе).

Свойство 4. Если код C' для X' оптимален, то код C оптимален для X.

Доказательство. Длины кодовых слов кодов C и C' по построению связаны соотношениями

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

Отсюда

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

$$= \sum_{m=1}^{M-2} p'_m l'_m + p'_{M-1} l'_{M-1} + p_{M-1} + p_M = \sum_{m=1}^{M-1} p'_m l'_m + p_{M-1} + p_M = \bar{l}' + p_{M-1} + p_M .$$

Последние два слагаемых в правой части не зависят от кода, поэтому код, минимизирующий $\bar{l}'$ одновременно обеспечивает минимум для $\bar{l}$.

Итак, сформулированные свойства оптимальных префиксных кодов сводят задачу построения кода объема M к задаче построения кодов объема $M' = M-1$. Это означает, что мы получили рекуррентное правило построения кодового дерева оптимального неравномерного кода.

Алгоритм построения кодового дерева кода Хаффмена.

1. Инициализация. Список «подлежащих обработке» узлов состоит из $M_0 = M$ узлов, каждому из которых приписана вероятность одного из сообщений.
2. До тех пор, пока число узлов $M_0 > 1$, повторяются следующие действия:

   В списке необработанных узлов находим два узла с наименьшими вероятностями. Они исключаются из списка и вместо них вводится новый узел, которому приписывается суммарная вероятность двух исключенных узлов. Новый узел связывается ребрами с исключенными узлами. Число необработанных узлов M_0 уменьшается на 1, $M_0 \leftarrow M_0 - 1$.

После выполнения алгоритма будет получено кодовое дерево, по которому можно построить код, который, как доказано выше, будет иметь наименьшую возможную среднюю длину кодовых слов.

Пример. 2.4.1. Рассмотрим ансамбль буквенных сообщений $X = \{a,b,c,d,e,f\}$ с вероятностями букв {0,35, 0,2, 0,15, 0,1, 0,1, 0,1} соответственно. Кодовое дерево и код Хаффмена показаны на рис. 2.4. Энтропия источника $H = 2{,}4016$. Средняя длина кодовых слов равна $\bar{l} = 2(0{,}35 + 0{,}2) + 3(0{,}15 + 3 * 0{,}1) = 2{,}45$. Согласно свойствам 1-4 не существует кода для X со средней длиной кодовых слов меньшей, чем 2,45.

2.5. Избыточность кода Хаффмена

Из теоремы 2.3.1 следует, что для построенных по алгоритму Хаффмена кодов средняя длина кодовых слов удовлетворяет неравенству

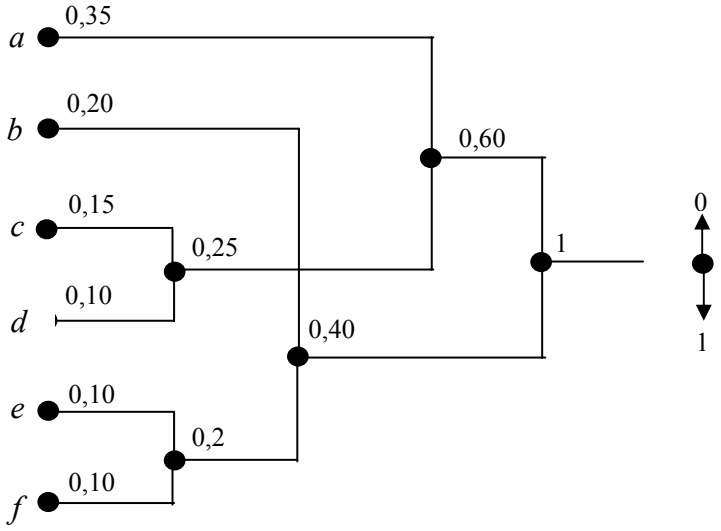
$$\bar{l} \le H + 1, \qquad (2.5.1)$$

где H – энтропия ансамбля.

Разность

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

называется *избыточностью* неравномерного кода. Она показывает степень «несовершенства» кода в том смысле, что при кодировании с избыточностью r на каждое сообщение тратится на r бит больше, чем в принципе можно было бы потратить, если использовать теоретически наилучший (возможно, нереализуемый) способ кодирования.



Буква	Кодовое слово
a	00
b	10
c	010
d	011
e	110
f	111

Рис. 2.5.1. Пример построения кода Хаффмена

Итак, из (2.5.1) следует, что для кода Хаффмена избыточность $r \le 1$. Однако, при решении практических задач, как и в примере 2.3, избыточность оказывается существенно меньше 1. Поэтому хотелось бы получить более точную оценку средней длины кодовых слов. Как уже говорилось, этого нельзя сделать, не ограничив множество рассматриваемых источников. Р. Галлагер в 1978 г. в работе, посвященной 25-летию кода Хаффмена, получил гораздо более точную оценку избыточности, наложив ограничение на максимальную из вероятностей сообщений. Сформулируем точно результат Галлагера в виде теоремы, которую мы приводим здесь без доказательства.

Теорема 2.5.1. Пусть p_1 - наибольшая из вероятностей сообщений конечного дискретного ансамбля. Тогда избыточность кода Хаффмена для этого ансамбля удовлетворяет неравенствам

$$r \le \begin{cases} p_1 + \sigma, & \text{при } p_1 < 1/2, \\ 2 - h(p_1) - p_1 & \text{при } p_1 \ge 1/2, \end{cases} \qquad (2.5.2)$$

где $h(x) = -x \log x - (1-x)\log(1-x)$ - энтропия двоичного ансамбля, и

$$\sigma = 1 - \log e - \log\log e \approx 0{,}08607\,. \qquad (2.5.3)$$

Отметим, что оценка Галлагера довольно точна. Для ансамбля с вероятностями сообщений (1/3, 1/3, 1/3, 0) имеем оценку 0,419. Истинная избыточность кода Хафмена для этого источника 0,417. Для троичного источника вида $(p_1, 1-p_1, 0)$ оценка теоремы дает точный результат. Эти примеры говорят о том, что существенно улучшить оценку, если известно только значение p_1, довольно трудно. В то же время, для примера 2.4 теорема 2.5.1 дает оценку $r \le 0{,}35 + 0{,}0861 = 0{,}4361$ при том, что истинное значение избыточности равно $r = 2{,}45 - 2{,}4016 = 0{,}0484$.

То, что оценка Галлагера неточна при $p_1 < 1/2$, стимулировало поиск более точных оценок в рамках той же постановки задачи. Было опубликовано довольно много работ на

эту тему и окончательный результат приведен в работе Манстеттена [Man92]. Им получена точная граница для всех значений p_1. Эта граница точна в том смысле, что она не может быть улучшена, если известно только значение p_1 максимальной вероятности сообщений источника. Полная запись формул Манстеттена заняла бы несколько страниц. Вместо этого на рис. 2.5.1 показаны графики оценок избыточности из работ Галлагера и Манстеттена как функций параметра p_1.

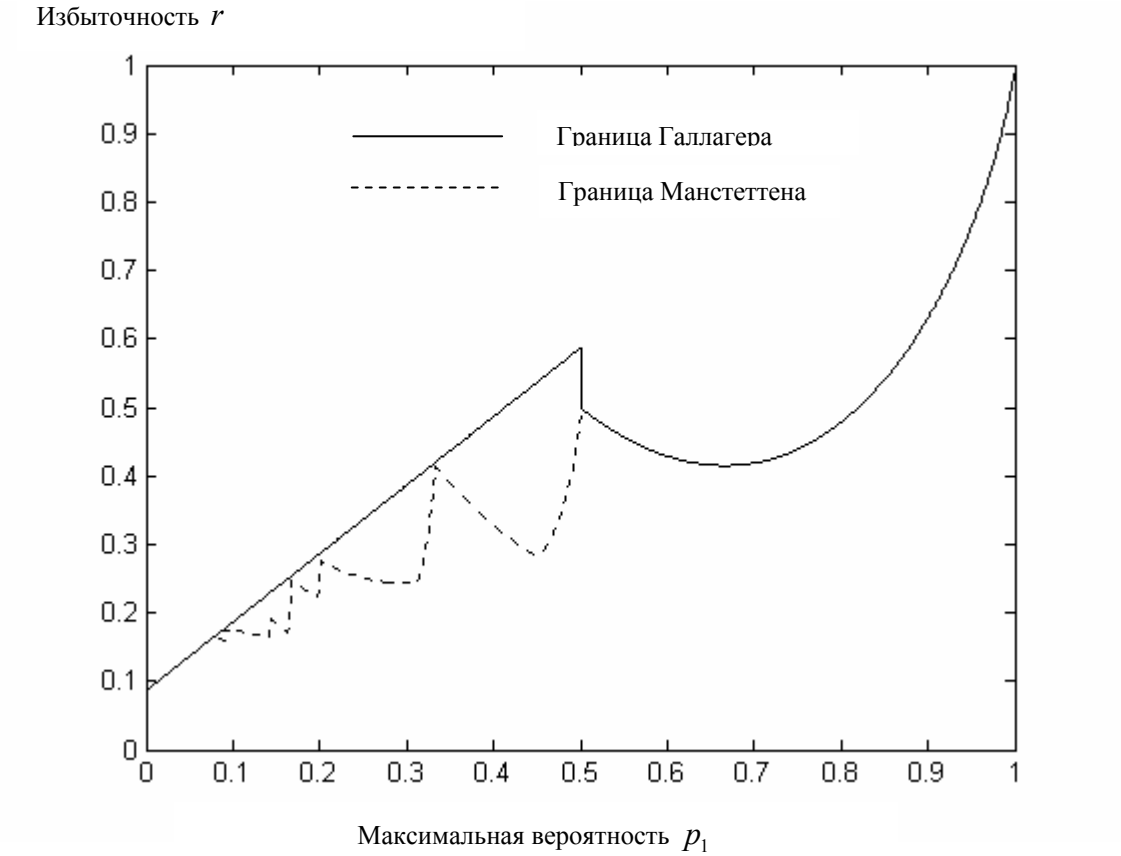


Рис. 2.5.1. Верхние оценки избыточности кода Хаффмена

2.6. Код Шеннона

После того, как мы научились строить оптимальные неравномерные коды, можно было бы закончить изучение побуквенного кодирования. Тем не менее, мы рассмотрим сначала код Шеннона, который заметно хуже кода Хаффмена, а затем изучим еще более плохие коды – коды Гилберта-Мура. В обоих случаях некоторое увеличение избыточности по сравнению с оптимальным кодом оправдывается простотой процедуры построения кодовых слов. При обсуждении прямой теоремы побуквенного кодирования уже говорилось о том, что избыточность 1 бит на букву не так уж велика, если велико значение энтропии. В следующем разделе мы перейдем к кодированию последовательностей сообщений. Целая последовательность будет рассматриваться как отдельное сообщение. Энтропия такого «расширенного» источника будет очень большой и главными проблемами будут уже не избыточность побуквенного кодирования, а сложность построения кодовых слов и сложность их декодирования.

Рассмотрим источник, выбирающий сообщения из множества $X = \{1,...,M\}$ с вероятностями $\{p_1,..., p_M\}$. Считаем, что буквы упорядочены по убыванию вероятностей,

т.е. $p_1 \geq p_2 \geq \ldots \geq p_M$. Сопоставим, кроме того, каждой букве так называемую *кумулятивную вероятность* по правилу $q_1 = 0$, $q_2 = p_1, \ldots, q_M = \sum_{i=1}^{M-1} p_i$.

Кодовым словом кода Шеннона для сообщения с номером m является двоичная последовательность, представляющая собой первые $l_m = \lceil -\log p_m \rceil$ разрядов после запятой в двоичной записи числа q_m.

Пример 2.6.1. Рассмотрим тот же ансамбль, что и в примере 2.4.1. В таблице 2.6.1 приведены промежуточные вычисления и результат построения кода Шеннона. Средняя длина кодовых слов $\bar{l} = 2{,}95$. В данном случае избыточность кода Шеннона оказалась на 0,5 бита больше, чем избыточность кода Хаффмена. Кодовое дерево кода показано на рис. 2.6.1. Из этого рисунка понятно, почему код неэффективен. Кодовые слова для букв b, d, e, f могут быть укорочены на 1 бит без потери свойства однозначной декодируемости.

Таблица 2.6.1

Построение кода Шеннона для примера 2.6.1.

Буква	Вероятность p_m	Кумулятивная вероятность q_m	Длина кодового слова l_m	Двоичная запись $[q]_2$	Кодовое слово $\mathbf{c}_m$
a	0,35	0,00	2	0,00…	00
b	0,20	0,35	3	0,0101…	010
c	0,15	0,55	3	0,10001…	100
d	0,10	0,70	4	0,10110…	1011
e	0,10	0,80	4	0,11001…	1100
f	0,10	0,90	4	0,11100…	1110

Докажем однозначную декодируемость кода Шеннона. Для этого выберем сообщения с номерами i и j, $i < j$. Кодовое слово $\mathbf{c}_i$ для i заведомо короче, чем слово $\mathbf{c}_j$ для j, поэтому достаточно доказать, что эти слова отличаются в одном из первых l_i символов. Рассмотрим разность

$$q_j - q_i = \sum_{k=1}^{j-1} p_k - \sum_{k=1}^{i-1} p_k = \sum_{k=i}^{j-1} p_k \geq p_i .$$

Вспомним, что длина слова и его вероятность связаны соотношением

$$l_i = \lceil -\log p_i \rceil \geq -\log p_i .$$

Поэтому

$$p_i \geq 2^{-l_i} .$$

С учетом этого неравенства

$$q_j - q_i \geq 2^{-l_i}.$$

В двоичной записи числа в правой части мы имеем после запятой $l_i - 1$ нулей и единицу в позиции с номером l_i. Это означает, что по меньшей мере в одном из l_i разрядов слова $\mathbf{c}_i$ и $\mathbf{c}_j$ отличаются и, следовательно, $\mathbf{c}_i$ не является префиксом для $\mathbf{c}_j$. Поскольку это верно для любой пары слов, то код является префиксным.

Заметим, что длины кодовых слов в коде Шеннона точно такие же, какие были выбраны при доказательстве прямой теоремы кодирования. Повторяя выкладки, получим уже известную оценку для средней длины кодовых слов

$$\bar{l} \le H + 1.$$

Примечательно, что при построении кода Шеннона мы выбрали длины кодовых слов приблизительно равными (чуть большими) собственной информации соответствующих сообщений. В результате средняя длина кодовых слов оказалось приблизительно равной (чуть большей) энтропии ансамбля.

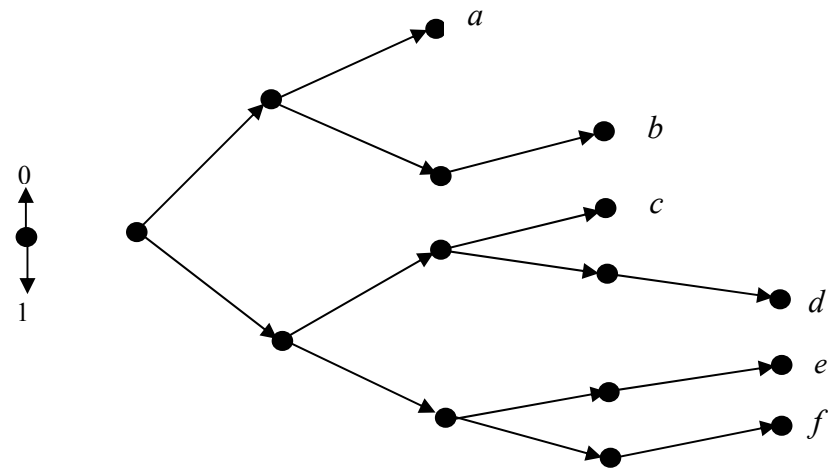


Рис. 2.6.1. Код Шеннона для источника из примера 2.6.1.

Хотя конструкция кода и анализ его характеристик уже достаточно понятны, мы обсудим еще одну, графическую, интерпретацию процесса кодирования. Она будет полезна в дальнейшем при обсуждении арифметического кодирования.

Рассмотрим числовой отрезок [0,1), на котором расположим один за другим отрезки длины $p_1,...,p_M$. На рис. 2.6.2 приведен пример разметки отрезка для случая $M = 3$, $p_1 = 0{,}6$, $p_2 = 0{,}3$, $p_3 = 0{,}1$. Как видно на рис. 2.6.2а, кумулятивные вероятности $q_1 = 0$, $q_2 = 0{,}6$, и $q_3 = 0{,}9$ соответствуют началам отрезков. Эти точки идентифицируют сообщения источника.

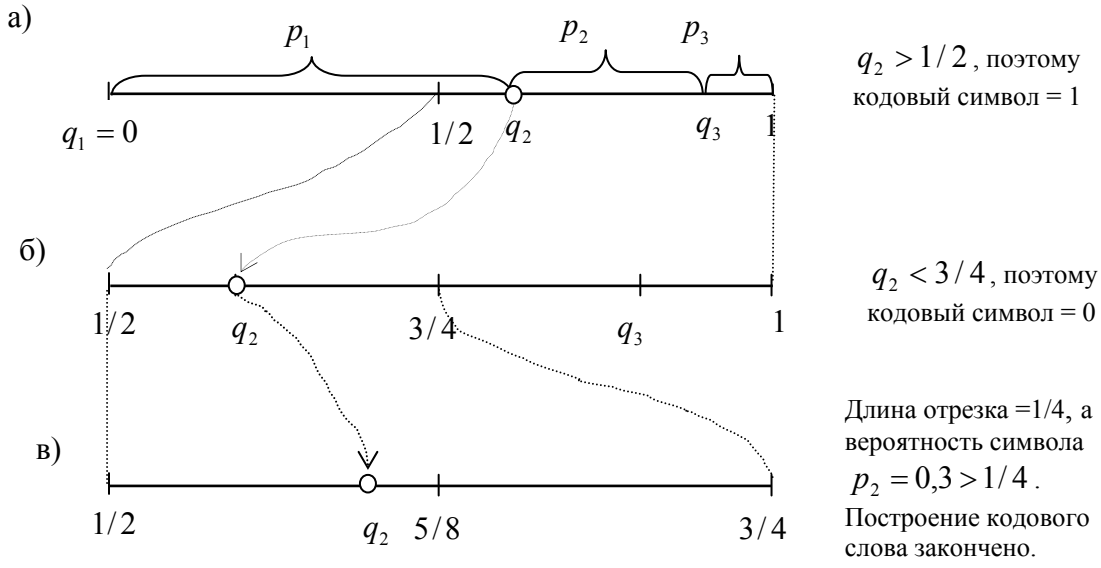


Рис. 2.6.2. Графическая интерпретация кода Шеннона

Предположим, что требуется закодировать сообщение с номером $m = 2$. Соответствующая ему точка помечена на рис. 2.6.2а-в. кружком. На первом шаге передачи передачей кодового символа 0 либо 1 кодер указывает, в какой половине отрезка [0,1} (левой или правой) находится начало соответствующего сообщению отрезка. В данном случае передается 1 и тем самым область возможных положений передаваемой точки уменьшается вдвое, что и показано на рис. 2.6.2б. На следующем шаге передается символ 0, т.к. точка находится в левой половине интервала и длина интервала неопределенности уменьшается до 1/4. Заметим, что после второго шага в интервале неопределенности осталась только одна точка и поэтому передача может быть завершена. Это произошло не случайно. Дело в том, что длина интервала равна 1/4, что меньше длины кратчайшего прилегающего отрезка 0,3. Именно поэтому гарантируется единственность точки и, значит, однозначность декодирования.

В общем случае после передачи l_m двоичных символов длина интервала неопределенности равна 2^{-l_m}. Декодирование будет однозначным, если $2^{-l_m} \le p_m$ или $l_m \ge -\log p_m$. Это условие в точности совпадает с правилом выбора длин кодовых слов в коде Шеннона.

Заметим, что при выборе длины кодовых слов мы ориентировались только на отрезок, лежащий справа от точки q_m. Упорядоченность букв по убыванию вероятностей гарантирует, что левый отрезок всегда будет длиннее правого.

2.7. Код Гилберта-Мура

При построении кода Шеннона мы требовали упорядоченности сообщений по убыванию вероятностей. В алгоритме построения кода Шеннона сортировка букв входного алфавита, пожалуй, наиболее трудоемкая часть. Мы заметно упростим построение кода, если модифицируем кодирование таким образом, чтобы упорядоченность не требовалась. Графическая интерпретация кода Шеннона, показанная на рис. 2.7, подсказывает почти очевидный путь к решению этой задачи.

Предположим, что вероятности не упорядочены и тогда при кодировании сообщения с номером m нужно учитывать не только вероятность (длину отрезка) p_m, но и длину предшествующего отрезка p_{m-1}, которая может быть очень маленькой, почти нулевой, и тогда длина слова будет большой даже если вероятность p_m велика. Как же избавиться от влияния p_{m-1}? Очень просто. Нужно соответствующую сообщению точку переместить из начала отрезка (точка q_m) в его середину (точка $q_m + p_m/2$), а длину кодового слова выбрать так, чтобы к концу передачи кодового слова длина интервала неопределенности была не больше $p_m/2$. Это и будет кодовое слово кода Гилберта-Мура!

Определим код Гилберта-Мура формально.

Рассмотрим источник, выбирающий буквы из алфавита $X = \{1,...,M\}$ с вероятностями $\{p_1,...,p_M\}$. Сопоставим, каждой букве $m = 1,...,M$ кумулятивную вероятность $q_m = \sum_{i=1}^{m-1} p_i$ и вычислим для каждой буквы величину σ_m по формуле

$$\sigma_m = q_m + \frac{p_m}{2}.$$

Кодовым словом кода Гилберта-Мура для x_m является двоичная последовательность, представляющая собой первые $l_m = \lceil -\log(p_m/2) \rceil$ разрядов после запятой в двоичной записи числа σ_m.

Пример 2.7.1. Рассмотрим источник с распределением вероятностей $p_1 = 0{,}1$, $p_2 = 0{,}6$, $p_3 = 0{,}3$. Вычисления, связанные с построением кода Гилберта-Мура для этого источника, приведены в таблице 2.2. Запись $[a]$ обозначает представление числа a в двоичной форме. В последнем столбце таблице показано, как выглядели бы кодовые слова соответствующего кода Шеннона, если бы мы забыли упорядочить буквы по вероятностям. Код, как и должно быть, получился непрефиксным.

Таблица 2.7.1

Построение кода Гилберта-Мура

Буква x_m	Вероятность p_m	Кумулятивная вероятность q_m	σ_m	Длина слова l_m	Кодовое слово кода Гилберта-Мура	Кодовое слово кода Шеннона (без упорядочения вероятностей)
1	0,1	0,0=[0,00000…]	0,05=[0,00001…]	5	00001	0000
2	0,6	0,1=[0,00011…]	0,40=[0,01100…]	2	01	0
3	0,3	0,7=[0,10110…]	0,85=[0,11011…]	3	110	10

Докажем однозначную декодируемость кода Гилберта-Мура в общем случае. Для этого выберем сообщения с номерами i и j, $i < j$. Понятно, что $\sigma_j > \sigma_i$. Нужно доказать, что соответствующие слова $\mathbf{c}_i$ и $\mathbf{c}_j$ отличаются хотя бы в одном из первых $\min\{l_i, l_j\}$ кодовых символов. Рассмотрим разность

$$\sigma_j - \sigma_i = \sum_{h=1}^{j-1} p_h - \sum_{h=1}^{i-1} p_h + \frac{p_j}{2} - \frac{p_i}{2} = \sum_{h=i}^{j-1} p_h + \frac{p_j - p_i}{2} \geq$$

$$\geq p_i + \frac{p_j - p_i}{2} = \frac{p_j + p_i}{2} \geq \frac{\max\{p_i, p_j\}}{2}.$$

Вспомним, что длина слова и его вероятность связаны соотношением

$$l_m = \left\lceil -\log \frac{p_m}{2} \right\rceil \geq -\log \frac{p_m}{2}.$$

Отсюда следует

$$\sigma_j - \sigma_i \geq \frac{\max\{p_i, p_j\}}{2} \geq 2^{-\min\{l_i, l_j\}},$$

а это означает, что слова $\mathbf{c}_i$ и $\mathbf{c}_j$ отличаются в одном из первых $\min\{l_i, l_j\}$ двоичных символов и поэтому ни одно из двух слов не может быть началом другого.

Поскольку все слова кода Гилберта-Мура не более чем на 1 длиннее слов кода Шеннона, получаем следующую оценку средней длины кодовых слов

$$\bar{l} \leq H + 2.$$

Мы завершим рассмотрение кода Гилберта-Мура графической интерпретацией, представленной на рис. 2.7.1 для источника из примера 2.7.1. Мы снова предполагаем, что передается буква с номером 2. Ей соответствует точка σ_2. По построению, соседние точки σ удалены от нее на расстояние по меньшей мере $p_2 / 2$. Кодер передает бит за битом и при этом каждый раз интервал неопределенности сужается вдвое. Передачу можно завершить, когда длина интервала неопределенности будет не больше $p_2 / 2$. В данном примере достаточно передать 2 бита.

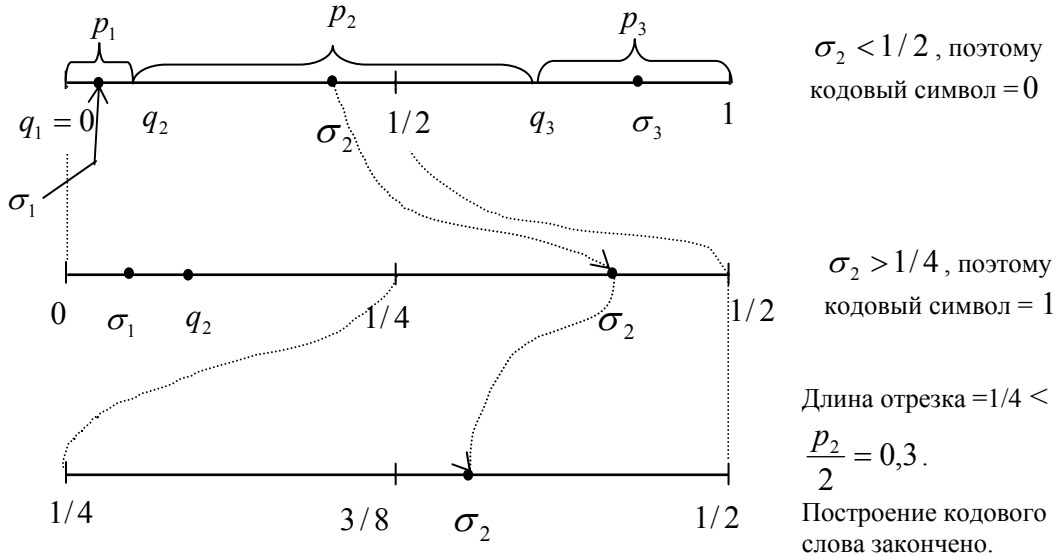


Рис. 2.7.1. Графическая интерпретация кода Гилберта-Мура

2.8. Неравномерное кодирование для стационарного источника. Постановка задачи. Теоремы кодирования

В этом параграфе мы перейдем к решению задачи кодирования последовательностей сообщений. Разумеется, если мы рассматриваем стационарный источник и его распределение вероятностей на буквах не меняется от букве к букве, то любой из описанных выше способов кодирования может быть использован для кодирования отдельных сообщений источника. Во многих случаях именно такой подход используется на практике как самый простой и достаточно эффективный.

В то же время, можно выделить класс ситуаций, когда побуквенное кодирование заведомо неоптимально. Во-первых, из теоремы об энтропии на сообщение стационарного источника следует, что учет памяти источника потенциально может значительно повысить эффективность кодирования. Во-вторых, побуквенные методы затрачивают как минимум 1 бит на сообщение, тогда как энтропия на сообщения может быть значительно меньше 1.

Итак, рассмотрим последовательность $x_1, x_2, \ldots,\ x_i \in X = \{x\}$, наблюдаемую на выходе дискретного стационарного источника, для которого известно вероятностное описание, т.е. можно вычислить все многомерные распределения вероятностей и по ним – энтропию на сообщение $H = H_\infty(X)$.

Пусть указан некоторый способ кодирования, который для любых n для каждой последовательности $\mathbf{x} \in X^n$ на выходе источника строит кодовое слово $\mathbf{c}(\mathbf{x})$ длины $l(\mathbf{x})$. Тогда *средняя скорость кодирования* определяется как

$$\overline{R} = \frac{1}{n}\mathbf{M}\big[l(\mathbf{x})\big] \text{ бит/сообщение источника.}$$

Такое кодирование в терминах, которые мы ввели в п. 1.9, называется FV-кодированием (fixed-to-variable), поскольку блоки из фиксированного числа сообщений n кодируются кодовыми словами переменной длины. Наша задача – связать достижимые значения средней скорости FV-кодирования с характеристиками источника, в частности, с его энтропией на сообщение H. Начнем с обратной теоремы кодирования.

Теорема 2.8.1. Для дискретного стационарного источника с энтропией на сообщение H для любого FV-кодирования имеет место неравенство

$$\overline{R} \geq H.$$

Доказательство. Рассмотрим множество X^n. К элементам этого множества применим теорему побуквенного кодирования. Получим

$$\mathbf{M}\big[l(\mathbf{x})\big] \geq H(X^n) = nH_n(X) \geq nH_\infty(X) = nH\,.$$

Здесь второе неравенство имеет место в силу того, что $H_n(X)$ не возрастает с увеличением n. Отсюда следует, что

$$\overline{R}_n \geq H$$

при любых n. Таким образом,

$$\overline{R} = \inf_n R_n \geq H\,,$$

что и требовалось доказать. ■

Почти также просто доказывается прямая теорема кодирования.

Теорема 2.8.2. Для дискретного стационарного источника с энтропией на сообщение H и для любого $\delta > 0$ существует способ неравномерного FV-кодирования такой, для которого

$$\overline{R} \leq H + \delta\,.$$

Доказательство. Воспользовавшись прямой теоремой побуквенного кодирования для ансамбля X^n, мы можем утверждать, что для некоторого побуквенного кода имеет место неравенство

$$\mathbf{M}\big[l(\mathbf{x})\big] \leq H(X^n) + 1 = nH_n(X) + 1. \qquad (2.8.1)$$

По определению предела числовой последовательности существует достаточно большое число n_1 такое, что при всех $n \geq n_1$ имеет место неравенство

$$\big|H_n(X) - H\big| \leq \frac{\delta}{2}\,,$$

из которого следует, что

$$H_n(X) \leq H + \frac{\delta}{2}\,,\ n > n_1\,. \qquad (2.8.2)$$

Найдем n_2 такое, что при $n \geq n_2$ имеет место неравенство

$$\frac{1}{n} \leq \frac{\delta}{2}\,. \qquad (2.8.3)$$

С учетом (2.8.2) и (2.8.3) из (2.8.1) получаем при $n \geq \max\{n_1, n_2\}$

$$\overline{R} = \inf_m \overline{R}_m \leq \overline{R}_n = \frac{\mathbf{M}\big[l(\mathbf{x})\big]}{n} \leq H_n(X) + \frac{1}{n} \leq H + \frac{\delta}{2} + \frac{\delta}{2} = H + \delta\,,$$

что и требовалось доказать. ■

Итак, выбрав достаточно большую длину блоков n и применив к блокам побуквенное кодирование, мы получим кодирование со средней скоростью

$$H \leq \overline{R} \leq H + o(n)\,,$$

где $o(n) \to 0$ при $n \to \infty$.

К сожалению, этот внешне оптимистический результат оказывается почти бесполезным при решении практических задач. Основное препятствие на пути его применения – это экспоненциальный рост сложности при увеличении длины блоков n. Поясним эту проблему следующим простым примером.

Предположим, что кодированию подлежат файлы, хранящиеся в памяти компьютера. Символы источника – байты и, значит, объем алфавита $|X| = 2^8 = 256$. При кодировании последовательностей длины $n = 2$ объем алфавита вырастает до $|X^2| = 2^{16} = 65536$. Далее при $n = 3$, 4,… объемы алфавита будут $2^{24} = 16777216$, $2^{32} = 4294967296$,… Понятно, что работать с кодами таких размеров невозможно.

Описываемый в следующем параграфе метод арифметического кодирования позволяет эффективно кодировать блоки длины n с избыточностью порядка $2/n$ и со сложностью

растущей только пропорционально квадрату длины блока n. За счет пренебрежимо малого проигрыша в скорости кода сложность может быть сделана даже линейной по длине кода. Неудивительно, что арифметическое кодирование все шире применяется в разнообразных системах обработки информации.

2.9. Арифметическое кодирование

По отношению к алгоритму арифметического кодирования нетривиальным является вопрос о том, кого считать его автором. Идею алгоритма приписывают Элайесу, ссылаясь на книгу Абрамсона [Abr63], опубликованную в 1963 г. До конца 70-х годов эта идея не была востребована. Несколько статей, опубликованных на рубеже 70-х и 80-х годов носили скорее теоретический характер. Большую роль в понимании как идеи алгоритма, так и возможности его использования для решения практических, задач сыграла статья Уиттена, Нила и Клири [WNC87]. Пожалуй, именно этих авторов нужно считать авторами алгоритма в его нынешней форме. В то же время, когда стал известен сам алгоритм, стало понятно, что он является почти тривиальным обобщением кода Шеннона на последовательности. Возможно, поэтому в книге Кавера и Томаса [CT91] арифметическое кодирование называется алгоритмом Шеннона-Фано-Элайеса.

Рассмотрим для простоты дискретный постоянный источник, выбирающий сообщения из множества $X = \{1,...,M\}$, с вероятностями $\{p_1,...,p_M\}$. Обозначим через $\{q_1,...,q_M\}$ кумулятивные вероятности сообщений. Наша задача состоит в кодировании последовательностей множества $X^n = \{\mathbf{x}\}$. При описании алгоритма кодирования мы будем использовать обозначение $\mathbf{x}_i^j$ для краткой записи подпоследовательности $(x_i,...,x_j)$ в последовательности $\mathbf{x} = (x_1,...,x_n)$.

Мы хотели бы применить к ансамблю $X^n = \{\mathbf{x}\}$ достаточно простой и эффективной побуквенный код. Упрощение состоит в том, ни кодер ни декодер не хранят и не строят всего множества из $|X^n|$ кодовых слов. Вместо этого при передаче конкретной последовательности $\mathbf{x}$ кодером вычисляется кодовое слово $\mathbf{c}(\mathbf{x})$ только для данной последовательности $\mathbf{x}$. Правило кодирования, конечно, известно декодеру и он восстанавливает $\mathbf{x}$ по $\mathbf{c}(\mathbf{x})$, не имея полного списка кодовых слов.

Возможными кандидатами на использование в такой схеме можно рассматривать код Шеннона и код Гилберта-Мура. Однако использование кода Шеннона предполагает упорядоченность сообщений по убыванию вероятностей. При больших n сложность упорядочения окажется недопустимо большой, поэтому единственным претендентом остается код Гилберта-Мура.

В соответствии с правилом построения кода Гилберта-Мура кодовое слова формируется по вероятности $p(\mathbf{x})$ и кумулятивной вероятности $q(\mathbf{x})$ как первые $l(\mathbf{x}) = \lceil -\log p(\mathbf{x}) + 1 \rceil$ разрядов после точки в двоичной записи числа $\sigma(\mathbf{x}) = q(\mathbf{x}) + p(\mathbf{x})/2$.

Для того, чтобы вычислить $q(\mathbf{x})$, надо условиться о некоторой нумерации последовательностей из X^n. Наиболее естественный способ нумерации последовательностей – использование лексикографической упорядоченности. Лексикографический порядок на последовательностях будет обозначаться знаком «$\prec$». Запись $\mathbf{y} \prec \mathbf{x}$ будет означать, что $\mathbf{y}$ лексикографически предшествует $\mathbf{x}$.

Понятие *лексикографического порядка* определяется следующим образом.

Для последовательностей длины 1 (для отдельных сообщений из X) мы считаем, что сообщение с меньшим номером предшествуют сообщению с большим номером. Если, например, элементы X – числа, то $x \prec x'$, если $x < x'$, $x, x' \in X$.

Для двух последовательностей $\mathbf{x} = (x_1,...,x_n)$, $\mathbf{y} = (y_1,...,y_n)$ обозначим через i наименьший индекс такой, что $x_i \neq y_i$. Тогда $\mathbf{y} \prec \mathbf{x}$, если $y_i \prec x_i$.

Нетрудно видеть, что лексикографический порядок – это порядок, который обычно используется при составлении словарей.

Итак, основная задача состоит в вычислении кумулятивной вероятности

$$q(\mathbf{x}) = \sum_{\mathbf{y} \prec \mathbf{x}} p(\mathbf{y}), \tag{2.9.1}$$

поскольку для источника без памяти вероятности последовательностей $p(\mathbf{x})$ вычисляются достаточно просто по формуле

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i).$$

Выведем рекуррентную формулу для вычисления $q(\mathbf{x})$. Для этого выразим вероятность $q(\mathbf{x}_1^n)$ через $q(\mathbf{x}_1^{n-1})$. Следующая цепочка элементарных преобразований показывает, как это можно сделать

$$q(\mathbf{x}_1^n) = \sum_{\mathbf{y}_1^n \prec \mathbf{x}_1^n} p(\mathbf{y}_1^n) = \sum_{\mathbf{y}_1^{n-1} \prec \mathbf{x}_1^{n-1}} \sum_{y_n} p(\mathbf{y}_1^{n-1} y_n) + \sum_{\mathbf{y}_1^{n-1} = \mathbf{x}_1^{n-1}} \sum_{y_n \prec x_n} p(\mathbf{y}_1^{n-1} y_n) =$$

$$= \sum_{\mathbf{y}_1^{n-1} \prec \mathbf{x}_1^{n-1}} p(\mathbf{y}_1^{n-1}) + \sum_{\mathbf{y}_1^{n-1} = \mathbf{x}_1^{n-1}} p(\mathbf{y}_1^{n-1}) \sum_{y_n \prec x_n} p(y_n) = q(\mathbf{x}_1^{n-1}) + p(\mathbf{x}_1^{n-1}) q(x_n),$$

где $q(x_n)$ обозначает кумулятивную вероятность символа x_n.

Подытожим наши выкладки в виде рекуррентных формул

$$\begin{aligned} q(\mathbf{x}_1^n) &= q(\mathbf{x}_1^{n-1}) + p(\mathbf{x}_1^{n-1}) q(x_n), \\ p(\mathbf{x}_1^n) &= p(\mathbf{x}_1^{n-1}) p(x_n). \end{aligned} \tag{2.9.2}$$

В этой рекурсии каждая пара значений $(q(\mathbf{x}_1^i), p(\mathbf{x}_1^i))$ используется ровно на одном шаге при вычислении следующей пары $(q(\mathbf{x}_1^{i+1}), p(\mathbf{x}_1^{i+1}))$. Поэтому при реализации арифметического кодирования вновь вычисленные значения записываются в те же ячейки памяти, в которых находились предыдущие значения. В приведенном ниже алгоритме кумулятивные вероятности $q(\mathbf{x}_1^i)$, $i = 1,2,...$ хранятся в виде переменной F, вероятности последовательностей $p(\mathbf{x}_1^i)$, $i = 1,2,...-$ в виде переменной G. Окончательная формулировка алгоритма может быть записана в следующем виде.

Алгоритм арифметического кодирования.

1. Инициализация. По вероятностям $\{p_1,..., p_M\}$ сообщений источника вычисляются кумулятивные вероятности:
   $q_1 = 0$, для $j = 2,..., M$, $q_j = q_{j-1} + p_{j-1}$.
   Устанавливаются начальные значения вспомогательных переменных $F = 0$, $G = 1$.
   Принимается от источника последовательность сообщений $\mathbf{x} = (x_1,..., x_n)$.
2. Для $i = 1,..., n$ выполняются следующие вычисления
   $F \leftarrow F + q(x_i)G,$
   $G \leftarrow p(x_i)G.$
3. Кодовое слово для $\mathbf{x}$ формируется как первые $\lceil -\log G \rceil + 1$ разрядов после запятой в двоичной записи числа $(F + G/2)$.

Пример 2.9.1. Рассмотрим источник из примера 2.7.1: $X = \{a,b,c\}$, распределение вероятностей $p_a = 0{,}1$, $p_b = 0{,}6$, $p_c = 0{,}3$. Вычисления, выполняемые арифметическим кодером при кодировании последовательности $\mathbf{x} = (bcbab)$ длины $n = 5$, приведены в

таблице 2.9.1. В этой таблице через $\hat{F}$ обозначено число $(F + G/2)$ округленное вниз с точностью до $\lceil -\log G + 1 \rceil = 9$ двоичных разрядов.

Таблица 2.9.1

Кодирование последовательности из примера 2.6 арифметическим кодом

Шаг i	x_i	$p(x_i)$	$q(x_i)$	F	G
0	-	-	-	0,0000	1,0000
1	b	0,6	0,1	0,1000	0,6000
2	c	0,3	0,7	0,5200	0,1800
3	b	0,6	0,1	0,5380	0,1080
4	a	0,1	0,0	0,5380	0,0108
5	b	0,6	0,1	0,5391	0,0065
6	Длина кодового слова		Кодовое слово		
	$\lceil -\log G + 1 \rceil = 9$		$F + G/2 = 0{,}5423... \rightarrow \hat{F} = 0{,}541 \rightarrow 100010101$		

Графическая интерпретация процесса кодирования аналогичная интерпретации кодов Шеннона и Гилберта-Мура показана на рис. 2.9.1.

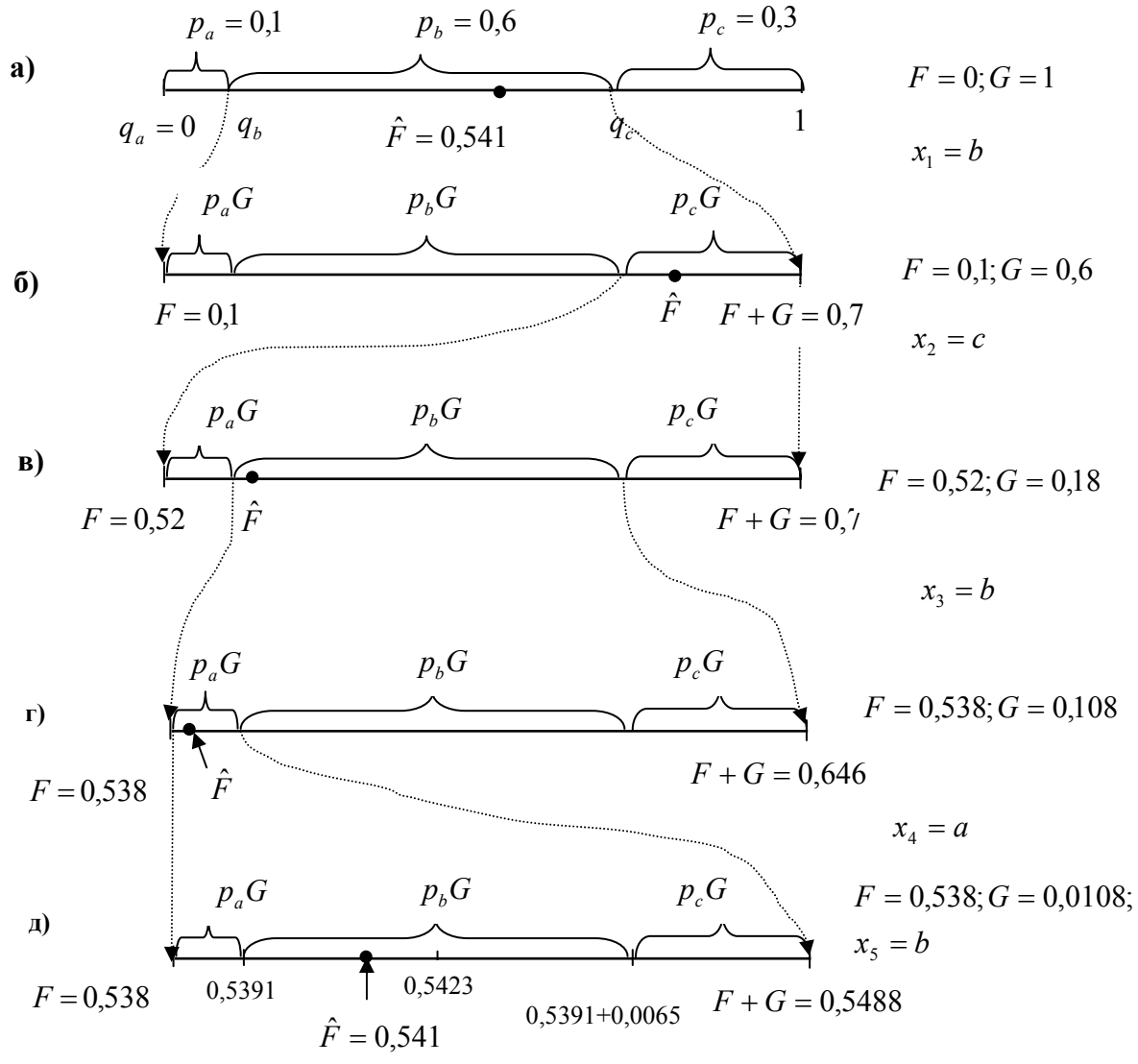


Рис. 2.9.1. Графическая интерпретация арифметического кодирования

Как показано на рисунке, на каждом шаге кодирования пересчитывается начальная точка F и длина G отрезка, в котором будет расположено число, соответствующее кодовой последовательности для заданной последовательности сообщений. Так, после первого шага ($x_1 = b$) мы знаем, что точка будет расположена в отрезке [0,1, 0,7). Более детально этот отрезок показан на рис. 2.9.1б. Поскольку вторая буква $x_2 = c$, начальная точка перемещается в значение 0,52, а длина интервала уменьшается до 0,18 и т.д. После 5-го шага $F = 0{,}5391$. Добавив смещение $G/2$ и округлив до 9 двоичных знаков, получаем величину $\hat{F} = 0{,}541$. Тем самым вся последовательность сообщений отображается в одну точку интервала [0,1). Эта точка помечена кружком на рисунках 2.9.1а – 2.9.1д. Как было показано выше, 9 разрядов достаточно для того, чтобы последовательность была восстановлена однозначно, т.е. ближайшая возможная точка, соответствующая другой последовательности сообщений, удалена от $\hat{F}$ на расстояние не менее $1/2^9 = 1/512$.

Обсудим кратко вопрос о сложности кодирования.

Из описания алгоритма следует, что на каждом шаге кодирования выполняется одно сложение и 2 умножения. Отсюда легко сделать неправильный вывод о том, что сложность кодирования последовательности из n сообщений пропорциональна n. Это неверно, поскольку на каждом шаге линейно растет сложность выполнения самих операций сложения и умножения, т.к. нарастает число двоичных разрядов, необходимых для записи операндов.

Предположим, что для представления вероятностей $p_1,...,p_M$ использованы числа разрядности d. После первого шага кодирования точное представление F и G потребует $2d$ разрядов, …, после n шагов кодирования кодер и декодер будут работать (в худшем случае) с числами разрядности nd, и, следовательно, суммарная сложность имеет порядок

$$d + 2d + ...nd = \frac{n(n+1)}{2} d.$$

Таким образом, можно говорить о том, что сложность арифметического кодирования пропорциональна n^2.

На самом деле, возможна практическая реализация арифметического кодирования со сложностью пропорциональной n. Вопросы реализации арифметического кодирования рассмотрены в следующем параграфе.

Отметим еще одну чрезвычайно важную особенность арифметического кодирования. Его легко адаптировать к случаю источников с памятью. Если, например, в качестве модели источника рассматривается простая цепь Маркова, то алгоритм кодирования остается прежним за тем исключением, что вместо одномерных вероятностей $p(x_i)$ и $q(x_i)$ нужно использовать условные вероятности $p(x_i \mid x_{i-1})$ и $q(x_i \mid x_{i-1}) = \sum_{y \prec x_i} p(x_i \mid x_{i-1})$.

2.10. Некоторые аспекты практической реализации арифметического кодирования

Как ясно из описания арифметического кодирования, при его практической реализации для кодирования последовательностей большой длины n возникают следующие проблемы:

- арифметическое кодирование требует большой (в пределе бесконечной) точности вычислений, что ведет к недопустимо высокой сложности реализации;

- для формирования кодового слова формально необходима вся последовательность сообщений, что приводит к недопустимо большой задержке кодирования, которая равна длине кодируемой последовательности сообщений.

Оказывается, что обе проблемы преодолимы. Решение состоит в том, что та часть данных, которая не участвует в дальнейших вычислениях и которая уже не влияет на окончательный результат, может быть исключена из вычислений и выдана на выход кодера. Тем самым уменьшается сложность вычислений и задержка кодирования. Чтобы пояснить, как это делается, рассмотрим работу декодера.

Начнем с анализа работы декодера кода Гилберта-Мура.

Пусть для источника $X = \{1,...,M\}$ известны вероятности $\{p_1,..., p_M\}$, по которым подсчитаны $q_m = \sum_{j=1}^{m-1} q_j$, $\sigma_m = q_m + p_m / 2$, длины слов $l_m = \lceil -\log(p_m / 2) \rceil$ и кодовые слова длины l_m, полученные округлением величин σ_m. Округленные до l_m разрядов после запятой числа σ_m обозначим через $\hat{\sigma}_m$. Задача декодера состоит в восстановлении сообщения m по соответствующему округленному значению $\hat{\sigma} = \hat{\sigma}_m$. Эта задача решается с помощью следующего алгоритма.

Алгоритм декодирования кода Гилберта-Мура .

1. Известны q_m, $m = 1,...,M$. Входом декодера служит число $\hat{\sigma}$.
   Полагаем $m = 1$.
2. До тех пор пока $q_{m+1} < \hat{\sigma}$ полагаем $m \leftarrow m + 1$.
3. Результатом декодирования является x_m.

Нетрудно проверить, что алгоритм работает правильно. Для этого достаточно вспомнить, что длина кодовых слов выбирается так, что в результате округления до l_m разрядов величины $\sigma_m = q_m + p_m / 2$ уменьшаются не более чем на $p_m / 2$ (погрешность округления не больше $2^{-l_m} \le p_m / 2$). Поэтому имеют место неравенства

$$q_m \le \hat{\sigma}_m < q_{m+1}.$$

В этом алгоритме при декодировании мы использовали только значения q_i, $i \le m$ и не использовали значений $\hat{\sigma}_i$. Это важно для применения к декодированию арифметического кода. Точно так же при декодировании арифметического кода для полученной из канала округленной величины $\hat{F}$ мы будем рекуррентно вычислять ближайшее к $\hat{F}$, но не превышающее $\hat{F}$ значение кумулятивной вероятности $q(\mathbf{x})$. Результатом декодирования будет соответствующая последовательность сообщений $\mathbf{x}$.

Итак, декодеру известны алфавит $X = \{1,...,M\}$, вероятности $\{p_1,..., p_M\}$, кумулятивные вероятности $\{q_1,...,q_M\}$, длина последовательности сообщений n и полученное из канала значение $\hat{F}$. Задача состоит в вычислении последовательности сообщений $\mathbf{x}$.

Алгоритм декодирования арифметического кода

1. Инициализируем начальные значение вспомогательных переменных $S = 0$, $G = 1$.
2. Для i от 1 до n выполняем следующие вычисления

   $j = 1$;

   До тех пор, пока $S + q_{j+1} G < \hat{F}$ полагаем $j \leftarrow j + 1$.

   После завершения цикла

   $S \leftarrow S + q_j G$;

   $G \leftarrow p_j G$;

   $x^{(i)} = j$.

В этом алгоритме переменная G на i-м шаге равна вероятности $p(\mathbf{x}_1^i)$ последовательности из первых i символов, а переменная S равна кумулятивной вероятности $q(\mathbf{x}_1^i)$.

Пример 2.10.1. Рассмотрим источник $X=\{a,b,c\}$ с распределением вероятностей $p_a=0{,}1$, $p_b=0{,}6$, $p_c=0{,}3$. Предположим, что на вход декодера поступила двоичная последовательность 0100010101. По этой последовательности нужно восстановить последовательность закодированных сообщений. Проще всего заглянуть на несколько страниц назад, ведь в примере 2.9.1 мы получили именно эту последовательность на выходе кодера. В таблице 2.10.1. приведены промежуточные результаты вычислений, выполненных при добросовестном использовании только что описанного алгоритма декодирования арифметического кода. Как видно из таблицы, последовательность сообщений восстановлена правильно.

Вернемся к вопросу о вычислительной сложности. Суть проблемы состоит в том, что во всех вычислениях как в кодере, так и декодере, мы выполняем вычисления над переменными, разрядность которых равна длине закодированной последовательности сообщений.

Таблица 2.10.1

Декодирование последовательности из примера 2.10.1

Шаг	S	G	Гипотеза x	$q(x)$	$S+qG$	Решение x_i	$p(x)$
$100010101\rightarrow 0,\ 100010101\rightarrow \hat{F}=0{,}541$							
1	0,0000	1,0000	a	0,0	$0{,}0000<\hat{F}$	b	0,6
			b	0,1	$\mathbf{0{,}1000}<\hat{F}$		
			c	0,7	$0{,}7000>\hat{F}$		
2	0,1000	0,6000	a	0,0	$0{,}1000<\hat{F}$	c	0,3
			b	0,1	$0{,}1600<\hat{F}$		
			c	0,7	$\mathbf{0{,}5200}<\hat{F}$		
3	0,5200	0,1800	a	0,0	$0{,}5200<\hat{F}$	b	0,6
			b	0,1	$\mathbf{0{,}5380}<\hat{F}$		
			c	0,7	$0{,}6460>\hat{F}$		
4	0,5380	0,1080	a	0,0	$\mathbf{0{,}5380}<\hat{F}$	a	0,1
			b	0,1	$0{,}5488>\hat{F}$		
5	0,5380	0,0108	a	0,0	$0{,}5380<\hat{F}$	b	0,6
			b	0,1	$\mathbf{0{,}5391}<\hat{F}$		
			c	0,7	$0{,}5456>\hat{F}$		

Рассмотрим подробнее работу кодера арифметического кода при кодировании последовательности из примера 2.9.1. Обратимся к таблице 2.9.1. Видно, что после 3-го шага кодирования значение F уже не станет меньше 0,5 и больше 0,75 независимо от того, какими будут последующие сообщения источника. Отсюда следует, что первые два символа числа F уже не изменятся и могут быть переданы по каналу. После этого можно

выполнить нормировку $F \leftarrow 2(F - 0{,}5)$, $G \leftarrow 2G$ и продолжить кодирование. Точно такую же нормировку выполнит и декодер. Тем самым разрядность переменных сократится без потери в точности вычислений. Нетрудно сформулировать общее правило выполнения такого рода нормировок. Трудность возникает в том случае, когда двоичное представление F содержит серию единиц после нуля (число F близко к 0,5, но меньше 0,5), т.е. $F = 0{,}01111...$. Неопределенность завершится одним из двух способов. Либо появится нулевой разряд ($F = 0{,}01111...10...$) либо произойдет перенос в одном из младших разрядов ($F = 0{,}10000...0...$). Если состояние неопределенности продлится долго, то разрядности ячеек, используемых для хранения F и G, может оказаться недостаточно. Выход из положения состоит в том, что кодер, встретившись с такой ситуацией, изменяет форму хранения числа F. Число разрядов уменьшается за счет того, что в памяти хранится не сама серия единиц, а ее длина. В момент появления нуля на выход выдается последовательность вида $0111...1$. Если же неопределенность завершилась переносом, то на выход поступает $1000..0$.

Похожие приемы применяются при реализации декодера. Кроме того, важно, что для вынесения решений относительно переданных символов не нужна вся последовательность $\hat{F}$, решения о сообщениях можно выдавать получателю, проанализировав часть кодовой последовательности. На примере 2.10.1 можно проследить, какой будет задержка принятия решений для каждого из 5 сообщений. Результаты вычислений приведены в таблице 2.10.2. Оказалось, что в данном случае первый символ можно выдать получателю, после получения из канала 2 двоичных символов. По первым 5 символам можно вычислить 3 первых сообщения, а для однозначного восстановления всех 5 сообщений достаточно 8 бит. Последний 9-й бит в данном случае оказался неиспользованным.

Таблица 2.10.2.

Анализ задержки декодирования для примера 2.7.

Шаг	Двоичный символ	Нижняя граница $\hat{F}$	Верхняя граница $\hat{F}$	Решение
1	1	0,5000	1,0000	-
2	0	0,5000	0,7500	*b*
3	0	0,5000	0,6250	
4	0	0,5000	0,5625	
5	1	0,5313	0,5625	*c* , *b*
6	0	0,5313	0,5469	*a*
7	1	0,5390	0,5469	
8	0	0,5390	0,5430	*b*
9	1	0,5410	0,5430	

Подробное описание алгоритма и текст С-программы арифметического кодера и декодера можно найти в статье [WNC87].

Раздел 3.

Кодирование дискретных источников при неизвестной статистике источника

Рассмотренные в предыдущем разделе алгоритмы кодирования эффективны и практичны. Однако, их общим недостатком является то, что они эффективны только в том случае, когда вероятностная модель источника совпадает или очень близка к модели, для которой строился код. В данном разделе мы рассмотрим задачу кодирования источника в условиях отсутствия информации или неполной информированности кодера и декодера о характеристиках источника. Методы кодирования для такой постановки задачи называют универсальными. Очевидная область применения универсального кодирования – создание архиваторов для хранения файлов в памяти ЭВМ. Эта задача служит полигоном для сопоставления эффективности различных подходов. В то же время имеются и другие важные приложения. Универсальные методы кодирования источников без потери качества являются важной составной частью современных кодеров аудио- и видеоинформации.

Мы начнем данный раздел с логически простых схем кодирования, на примере которых мы убедимся в том, что скорость кодирования при неизвестной статистике может быть сделана сколь угодно близкой к скорости создания информации источником. Мы оценим потери, связанные с отсутствием априорной информации. Практическая значимость изучаемых методов проявится в следующем разделе, где мы рассмотрим широко применяемые в известных архиваторах алгоритмы Зива-Лемпела, а также алгоритмы, обеспечивающие рекордно высокое сжатие информации – это предсказание по частичному совпадению и алгоритм Барроуза-Виллера.

3.1. Постановка задачи универсального кодирования источников

Универсальное кодирование предполагает, что входом кодера является последовательность сообщений $\mathbf{x} = (x_1,...,x_n)$ некоторого источника X. Алфавит кодера, длина последовательности n, алгоритм работы кодера известны декодеру. Статистические свойства источника заранее неизвестны. Задача, как и прежде, состоит в том, чтобы обеспечить эффективное кодирование, то есть найти представление последовательности источника двоичной кодовой последовательностью наименьшей длины. Однако, решить задачу в такой постановке нам не удастся. Не может существовать алгоритма кодирования, который бы «сжимал» любые последовательности источника. Мы уже говорили (см. п. 2.2), что при неравномерном префиксном кодировании уменьшение длины одной кодовой последовательности на 1 символ достигается увеличением длины двух других последовательностей на 1 символ. Если считать все последовательности источника одинаково вероятными, то любой алгоритм кодирования «в среднем» будет проигрывать равномерному кодированию, затрачивающему на каждое сообщение $\log |X|$ двоичных символов.

Мы существенно упростим ситуацию, если примем предположение о том, что источник сообщений является стационарным. В этом случае мы можем подсчитать для него энтропию на сообщение и тем самым определить минимально достижимую скорость кодирования (понятно, что обратная теорема кодирования остается справедливой при отсутствии у кодера и декодера информации о характеристиках источника). Для стационарного источника естественно считать «хорошим» тот способ кодирования, который обеспечивает скорость близкую к той скорости кодирования, которая могла бы быть достигнута при известной статистике.

Заметим, что при решении многих практических задач есть возможность дополнительно сузить класс возможных моделей источника, то есть помимо предположения о стационарности сделать и другие предположения. Разумеется, любая априорная информация может быть использована для повышения эффективности кодирования.

Опираясь на эти практические соображения, мы приходим к следующей постановке задачи. Пусть $\Omega = \{w\}$ представляет собой некоторый класс (множество) моделей источников (не обязательно дискретное). Например, в качестве Ω может рассматриваться множество дискретных постоянных источников. В этом случае конкретный элемент множества определяется одномерным распределением вероятностей на X. Обозначим через H_w энтропию на сообщение источника для заданной модели $w \in \Omega$. Пусть заданный алгоритм кодирования обеспечивает для этой модели среднюю скорость $\overline{R}_w$. Тем самым определена избыточность $r_w = R_w - H_w$. *Избыточностью алгоритма для класса моделей* Ω называется величина

$$r(\Omega) = \sup_{\varpi \in \Omega} [R_w - H_w].$$

Задача состоит в построении алгоритма, минимизирующего избыточность $r(\Omega)$ для заданного класса моделей Ω. При этом помимо ограничений на множество моделей источников возможны ограничения на множество допустимых алгоритмов. Мы отметим два наиболее важных ограничения.

Во-первых, во многих практических задачах накладывается ограничение на допустимую задержку кодирования. Примерами могут служить кодер, предназначенный для сжатия файлов в реальном времени в процессе записи информации на жесткий диск или кодер, входящий в состав модема и сжимающий информацию перед передачей по линии связи. С другой стороны, можно указать широкий круг задач, в которых задержка кодирования не имеет значения. Например, для производителя программного обеспечения при записи программ и данных на диск задержка кодирования не имеет решающего значения.

По этому критерию различают две крайние ситуации. Один класс алгоритмов рассчитан на применение в тех случаях, когда задержку недопустима. На английском языке такие алгоритмы называют on-line алгоритмами. Мы будем называть их кодированием без задержки или мгновенным кодированием. Общим свойством таких алгоритмов является то, что при кодировании каждой вновь поступившей буквы источника разрешается использовать информацию о предшествующих буквах, но не разрешается использовать информацию о будущих буквах.

Второй класс алгоритмов – алгоритмы для тех задач, в которых задержка не имеет значения. Эти алгоритмы на английском языке называются off-line алгоритмами. Мы будем использовать термин «двухпроходное кодирование». Этот термин правильно отражает суть кодирования при неограниченной задержке. Кодер сначала исследует статистические свойства последовательности и принимает решение о стратегии кодирования. Затем, на втором проходе, производится собственно кодирование.

Поскольку любой алгоритм мгновенного кодирования можно использовать в системах с неограниченной задержкой, второй класс алгоритмов, конечно, полностью включает первый класс алгоритмов. Вопрос, на который нам предстоит ответить – насколько велик выигрыш, который можно получить, «заглядывая вперед» при кодировании.

Еще одно ограничение, учитываемое при выборе алгоритма – сложность кодирования и декодирования.

Итак, цель исследования алгоритмов универсального кодирования – построение кодеров, обеспечивающих минимальную избыточность для заданного класса моделей при ограничениях на задержку и сложность.

3.2. Несколько полезных комбинаторных формул

В этом разделе нам постоянно придется подсчитывать число последовательностей, удовлетворяющих тем или иным свойствам. В связи с эти мы посвятим отдельный параграф самым необходимым комбинаторным формулам.

Начнем с того, что рассмотрим множество последовательностей вида $\mathbf{x} = (x_1,...,x_n)$, в которых элемент x_i может принимать одно из M_i значений, $i = 1,...,n$. Очевидно, число различных $\mathbf{x}$ равно

$$\left|\left\{\mathbf{x} = (x_1,...,x_n) : x_i \in \{0,...,M_i - 1\}, i = 1,...n\right\}\right| = \prod_{i=1}^{n} M_i \,. \tag{3.2.1}$$

В частности, число различных M -ичных последовательностей длины n равно M^n.

Предположим, что все компоненты $\mathbf{x} = (x_1,...,x_n)$ выбираются из одного и того же множества объема M, но «без возвращения», т.е. один и тот же элемент не может повториться в $\mathbf{x}$ больше одного раза. Число таких последовательностей представляет собой число *размещений* из M элементов по n и равно

$$A_M^n = M(M-1)\times...\times(M-n+1) = \frac{M!}{(M-n)!} \,. \tag{3.2.2}$$

Эта формула следует из (3.2.1). В частном случае, когда $M = n$, речь идет о всевозможных *перестановках* из M элементов. Число различных перестановок P_M равно

$$P_M = M! \tag{3.2.3}$$

Итак, существует A_M^n различных слов длины n из неповторяющихся букв алфавита объема M. При этом подразумевается, что порядок следования букв в словах существенен, т.е. слова, отличающиеся порядком букв различны. Подсчитаем теперь, сколько можно построить различных неупорядоченных подмножеств объема n из алфавита объема M. Поскольку $P_n = n!$ различных последовательностей образуют теперь одно подмножество, число различных подмножеств равно

$$C_M^n = \binom{M}{n} = \frac{A_M^n}{P_n} = \frac{M(M-1)\times...\times(M-n+1)}{n!} = \frac{M!}{n!(M-n)!}.. \tag{3.2.4}$$

Это число называют числом *сочетаний* из M элементов по n. Здесь мы привели два обозначения для числа сочетаний. Первое, C_M^n, распространено в российской математической литературе. Второе, $\binom{M}{n}$ используют в международных математических и технических изданиях. Мы будем придерживаться второго обозначения. Формула (3.2.4) не имеет смысла при $M, n \le 0$ и при $M < n$. Для всех этих случаев мы формально определим значения числа сочетаний равными нулю, т.е.

$$\binom{M}{n} = 0, \text{ если } M \le 0, \text{ или } n \le 0, \text{ или } M < n \,. \tag{3.2.4а}$$

Числа $\binom{M}{n}$ называют биномиальными коэффициентами в связи с использованием в известной формуле бинома Ньютона

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i} \,.$$

Подсчитаем число двоичных последовательностей длины n, содержащих заданное число τ_1 единиц и $\tau_0 = n - \tau_1$ нулей. Заметим, что номера позиций единиц образуют неупорядоченное подмножество множества чисел $\{1,...,n\}$. Из (3.4) следует, что искомое

число последовательностей равно

$$N(\tau_0,\tau_1)=\binom{n}{\tau_1}=\frac{n}{\tau_0!\tau_1!}. \qquad (3.2.5)$$

Обобщим эту формулу на множество последовательностей над произвольным алфавитом $X=\{0,...,M-1\}$. Напомним, что композицией последовательности $\mathbf{x}$ мы называем вектор $\boldsymbol{\tau}(\mathbf{x})=(\tau_0(\mathbf{x}),...,\tau_{M-1}(\mathbf{x}))$, в котором $\tau_i(\mathbf{x})$ обозначает число элементов $x_t=i$ в последовательности $\mathbf{x}=(x_1,...,x_n)$. Ближайшая цель состоит в нахождении числа последовательностей $\mathbf{x}$ с заданной композицией $\boldsymbol{\tau}=(\tau_0,...,\tau_{M-1})$.

Начнем с того, что положим $M=3$ и найдем число последовательностей длины n с композицией $\boldsymbol{\tau}=(\tau_0,\tau_1,\tau_2)$, $n=\tau_0+\tau_1+\tau_2$. Для этого зафиксируем некоторое расположение нулей (это можно сделать $\binom{n}{\tau_0}$ способами) и подсчитаем число различных расположений τ_1 единиц на оставшихся $n-\tau_0$ позициях. Это число, как мы знаем, равно $\binom{n-\tau_0}{\tau_1}$. Поскольку каждому расположению нулей соответствует такое количество расположений единиц, общее число последовательностей равно

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

Аналогично для алфавита произвольного объема M легко вывести формулу

$$N(\boldsymbol{\tau})=\frac{n!}{\tau_0!...\tau_{M-1}!}. \qquad (3.2.6)$$

Эта формула является естественным обобщением формулы для биномиальных коэффициентов. Также естественно обобщается формула бинома Ньютона на случай возведения в степень произвольного числа слагаемых:

$$(a_0+...+a_{M-1})^n=\sum_{\boldsymbol{\tau}:\tau_0+...+\tau_{M-1}=n}N(\boldsymbol{\tau})\prod_{i=0}^{M-1}a_i^{\tau_i}.$$

Решим еще одну комбинаторную задачу. Подсчитаем число различных композиций для последовательностей длины n над алфавитом объема M. Заметим, что каждая композиция это одно из представлений числа n в виде суммы M слагаемых. Сколько всего таких представлений? Ответ получается довольно просто.

Лемма 3.2.1. Натуральное число n может быть представлено в виде суммы M неотрицательных целых слагаемых $\binom{n+M-1}{M-1}$ способами.

Доказательство. Будем записывать последовательность из j нулей в виде 0^j. Каждому разбиению числа n на целые слагаемые $a_1+..+a_M=n$ соответствует двоичная последовательность $(0^{a_1}1,0^{a_2}1,...0^{a_M})$. Число разбиений равно числу таких последовательностей, которое, в свою очередь, равно числу двоичных последовательностей длины $n+M-1$, содержащих $M-1$ единиц. Это число как раз равно $\binom{n+M-1}{M-1}$.■

Из леммы 3.2.1 следует, что число различных композиций для последовательностей длины n над алфавитом объема M равно

$$N_\tau(n,M)=\binom{n+M-1}{M-1}. \qquad (3.2.7)$$

Полученные формулы достаточно просты для подсчетов с помощью компьютера, но неудобны для анализа. Ниже мы получим асимптотические формулы достаточно точные при больших длинах последовательностей n.

Все вычисления основаны на формуле Стирлинга

$$\sqrt{2\pi n}n^n e^{-n}\exp\left\{\frac{1}{12n+1}\right\} < n! < \sqrt{2\pi n}n^n e^{-n}\exp\left\{\frac{1}{12n}\right\}. \qquad (3.2.8)$$

Ее подстановка в (3.2.6) дает

$$N(\boldsymbol{\tau}) < (2\pi n)^{-\frac{M-1}{2}} 2^{n\log n-\sum_i \tau_i \log \tau_i}\left(\prod_i \frac{n}{\tau_i}\right)^{1/2}\exp\left\{\frac{1}{12n}-\sum_i \frac{1}{12\tau_i+1}\right\}. \qquad (3.2.9)$$

Из этой формулы получаем верхнюю оценку для логарифма числа последовательностей с заданной композицией

$$\log N(\boldsymbol{\tau}) < nH(\hat{\mathbf{p}}) - \frac{M-1}{2}\log(2\pi n) - \frac{1}{2}\sum_i \log(\hat{p}_i), \qquad (3.2.10)$$

где в (3.2.8) – (3.2.10) индекс i пробегает те значения, для которых τ_i положительно, через $\hat{p}_i$ обозначены вычисленные по $\mathbf{x}$ оценки вероятностей букв, т.е. $\hat{p}_i = \tau_i / n$, через $H(\hat{\mathbf{p}})$ обозначена энтропия ансамбля X, вычисленная по распределению вероятностей $\hat{\mathbf{p}} = (\hat{p}_0,...,\hat{p}_{M-1})$.

Помимо (3.2.10) можно также записать менее точную, но более компактную оценку

$$\log N(\boldsymbol{\tau}) < nH(\hat{\mathbf{p}}) - \frac{M-1}{2}\log(2\pi) + \frac{1}{2}\log\frac{n}{n-M+1}. \qquad (3.2.11)$$

В заключение сформулируем некоторые свойства биномиальных коэффициентов. Напомним, что число $\binom{n}{w}$ можно интерпретировать как число двоичных последовательностей длины n веса w. Имеет место рекуррентная формула

$$\binom{n+1}{w} = \binom{n}{w} + \binom{n}{w-1}. \qquad (3.2.12)$$

Действительно, любая последовательность длины $n+1$ веса w может быть получена либо из последовательности длины n веса w дописыванием нуля, либо из последовательности длины n веса $w-1$ дописыванием единицы. Можно теперь применить эту же формулу ко второму слагаемому в (3.12). Проделав это многократно, получим тождество

$$\binom{n+1}{w} = \binom{n}{w} + \binom{n-1}{w-1} + ... + \binom{n-w+1}{1} + 1. \qquad (3.2.13)$$

3.3. Двухпроходное побуквенное кодирование

Без потери общности будем считать, что источник выбирает сообщения из множества $X = \{0,...,M-1\}$. Пусть $\mathbf{x} = (x_1,...,x_n)$ – последовательность на выходе источника. Множество сообщений X, и длину последовательности считаем заранее известными кодеру и декодеру. На практике длина последовательности либо передается отдельно в заголовке файла некоторым стандартным для данной системы способом, либо в алфавите источника имеется специальный символ, указывающий на завершение файла. В любом случае для всех обсуждаемых ниже методов универсального кодирования проблема конца файла решается одинаковым способом и от способа ее решения сравнительные характеристики алгоритмов не зависят.

Для простоты рассмотрим универсальное кодирование для класса источников без памяти с неизвестным распределением вероятностей на буквах источника. Рассмотрим следующий очевидный способ решения задачи. Кодер сначала просматривает

последовательность и определяет число появлений $\tau_n(a)$ каждого сообщения $a \in X$ в последовательности $\mathbf{x}$ длины n. Затем, используя полученную информацию, вычисляет оценки вероятностей сообщений и строит по ним некоторый код для множества X. На втором проходе этот код используется для кодирования последовательности сообщений источника. Кодовое слово состоит из двух частей $\mathbf{c}(\mathbf{x}) = \{\mathbf{c}_1(\mathbf{x}), \mathbf{c}_2(\mathbf{x})\}$. Первая часть содержит информацию об использованном коде, вторая – собственно закодированную последовательность.

Декодер сначала по $\mathbf{c}_1(\mathbf{x})$ строит код, затем по $\mathbf{c}_2(\mathbf{x})$ восстанавливает одно за другим закодированные сообщения.

В это описание вкладывается целое семейство алгоритмов, отличающихся друг от друга выбором кода и способом передачи служебной информации. Мы рассмотрим два подхода, один алгоритм, обсуждаемый в данном подразделе, основан на побуквенное кодирование кодом Хаффмена. В следующем параграфе будет рассмотрен алгоритм, основанный на арифметическом кодировании.

Поясним работу двухпроходного побуквенного кодера примером.

Пример 3.3.1. Предположим**,** что алфавитом источника являются данные, хранящиеся в памяти ЭВМ в виде байтов, то есть объем алфавита $|X| = 256$. В качестве тестовой последовательности, на которой мы будем испытывать все рассматриваемые в этом разделе методы, мы выбрали поучительную пословицу

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

Мы заменили пробелы подчеркиванием, поскольку так удобнее отображать буквы алфавита при описании кодов. Результаты статистического анализа файла и соответствующий код Хаффмена для букв алфавита X приведены в таблице 3.3.1. Обозначим через l_1 и l_2 длину первой и второй части кодового слова. Длина l_2 равна сумме длин кодовых слов информационной последовательности. Подсчитать ее легко:

$$l_2 = 6 + 6 + 12 \times 2 + 5 \times 3 + \ldots + 6 = 178.$$

Чтобы подсчитать l_1 нужно условиться о способе передачи информации о коде. Первое, что приходит на ум – перечислить буквы, длины кодовых слов, сами кодовые слова. Этот подход заведомо неэффективен. Гораздо экономичнее передать структуру кодового дерева. Зная ее, декодер сам единственным образом восстановит все кодовые последовательности.

Кодовое дерево для рассматриваемого кода показано на рис. 3.3.1. Все вершины дерева размечены. Нулем помечены промежуточные вершины, единицей – концевые. Будем считывать символы, приписанные вершинам ярус за ярусом, начиная с корня дерева, сверху вниз. В данном случае мы получим двоичную последовательность вида

00010000010100110111101101111.

Эта последовательность из 29 двоичных символов не только полностью описывает дерево, но и несет информацию о количестве различных букв в последовательности. Действительно, декодер по этой последовательности ярус за ярусом может восстановить дерево. Прочитав первый нулевой символ, он узнает что последовательность содержала по меньшей мере 2 различные буквы. По следующим двум символам, он определяет, что на первом ярусе концевых узлов нет и, следовательно, на следующем (втором) ярусе имеется 4 вершины. Прочитав 4 символа, получаем концевую вершину на втором ярусе и 6 вершин на 3-м ярусе и т.д. На 6-м ярусе все вершины оказались концевыми, значит построение дерева завершено.

Таблица 3.3.1.

Код Хаффмена для примера 3.3.1.

Буква	Число появлений	Длина кодового слова	Кодовое слово
I	1	6	010000
F	1	6	010001
_	12	2	00
W	5	3	100
E	4	4	0101
C	2	5	01001
A	4	4	1010
N	3	4	1011
O	5	3	110
T	1	6	011110
D	4	4	0110
S	3	4	1110
U	2	4	1111
L	2	5	01110
H	1	6	011111

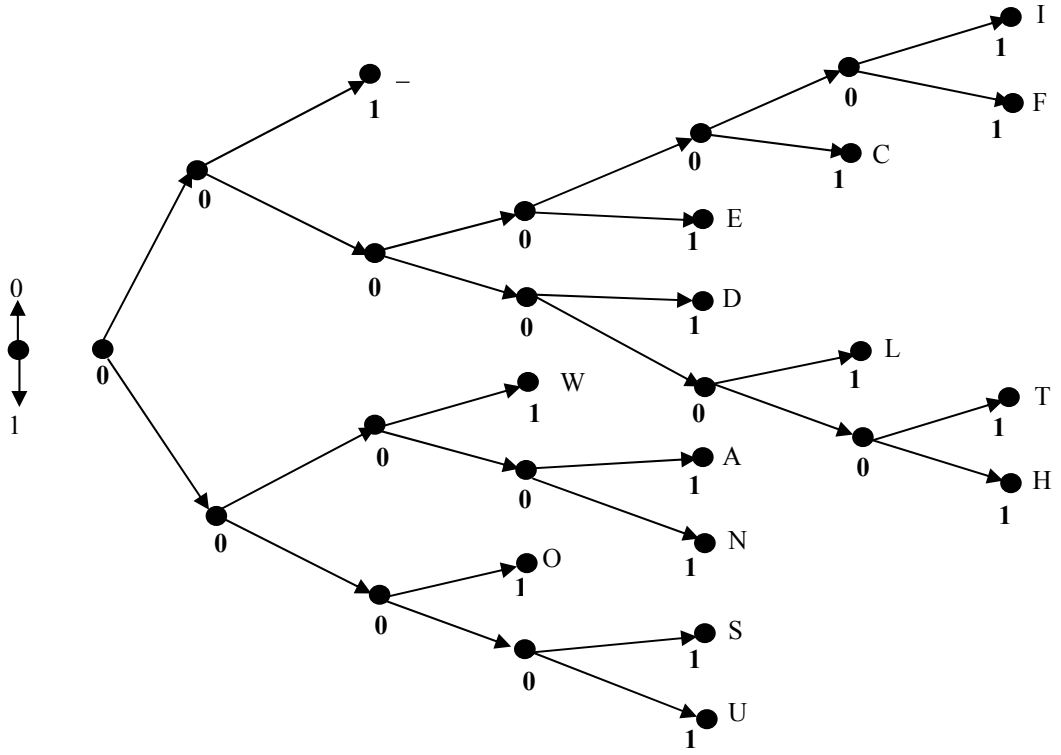


Рис. 3.3.1. Кодовое дерево для кода Хаффмена из примера 3.3.1.

Чтобы завершить описание кода, нужно указать, какая конкретно буква соответствует каждому кодовому слову. Для этого достаточно 8 бит на каждую букву. В результате затраты на информацию о коде составят

$$l_1 = 29 + 8 \times 15 = 149 \text{ бит.}$$

Передача всего файла потребует

$$l = l_1 + l_2 = 149 + 178 = 327 \text{ бит.} \qquad (3.3.1)$$

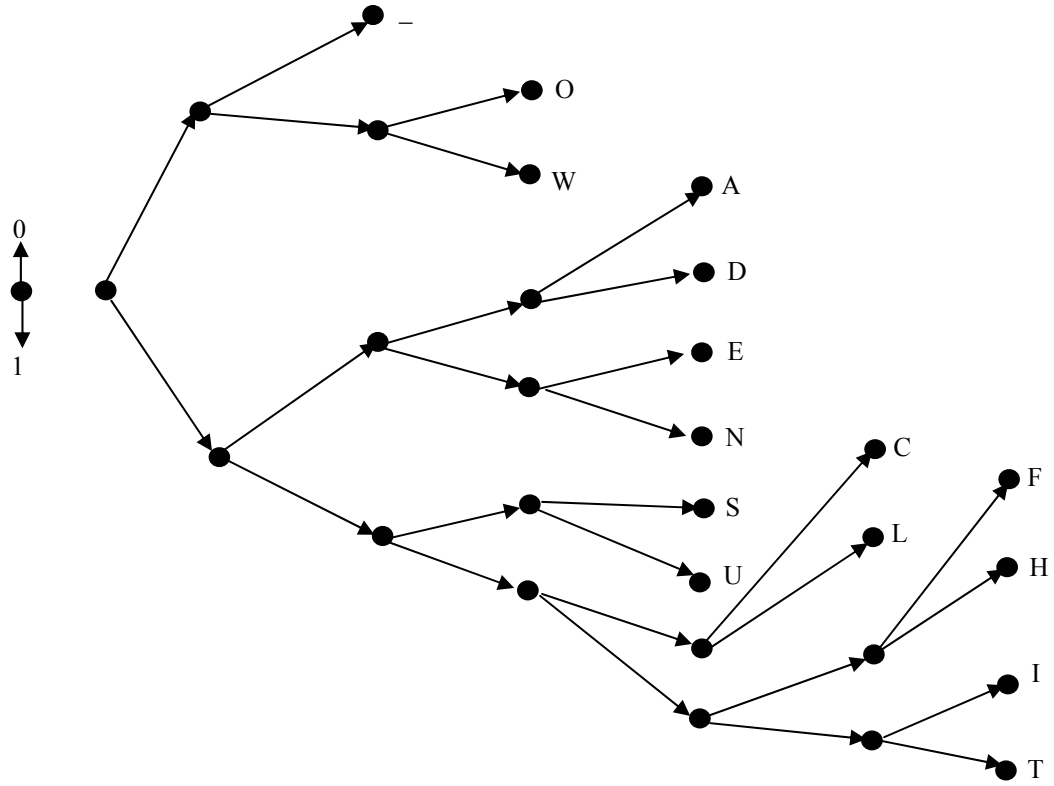


Рис. 3.3.2. Кодовое дерево регулярного кода Хаффмена для примера 3.3.1.

Отметим, что без кодирования мы затратили бы $50 \times 8 = 400$ бит. Обращает на себя внимание тот факт, что «вспомогательная» информация» (информация о коде) составляет почти половину длины кодовой последовательности. Это наводит на мысль о том, что с увеличением длины кодируемой последовательности доля служебной информации уменьшится и эффективность кодирования существенно возрастет. ■

Затраты на служебную информацию можно уменьшить, если заметить, что для одного и того же распределения вероятностей можно построить много кодов Хаффмена. Все эти коды одинаково эффективны. Нет необходимости передавать точную структуру дерева. Нужно передать минимальные сведения, достаточные для построения одного из кодов Хаффмена.

Назовем *регулярным* неравномерный код в котором короткие кодовые слова лексикографически предшествуют более длинным. Регулярный код для рассматриваемого примера приведен в таблице 3.3.2, а его дерево - на рис. 3.3.2. Для описания регулярного кода достаточно указать число концевых вершин на каждом из ярусов с номерами $0,\ldots,l_{\max}$, где $l_{\max}$ - максимальная длина кодового слова. Подсчет числа бит на передачу дерева для регулярного кода иллюстрируется таблицей 3.3.

Для передачи описания регулярного кода оказалось достаточно 19 бит. Дополнительный выигрыш можно получить, если учесть, что нет необходимости тратить 8 бит на указание того, какие буквы соответствуют каждому слову регулярного кода. Достаточно указать длину слова для каждой буквы алфавита. Кодер и декодер, зная состав множества слов одинаковой длины, одинаково упорядочат их внутри множества, например, в алфавитном порядке.

Таблица 3.3.2

Регулярный код Хаффмена для примера 3.3.1.

Буква	Номер буквы в алфавите	Номер яруса (длина кодового слова)	Кодовое слово	Затраты в битах на передачу номера буквы в алфавите
_	32	2	00	8
O	15	3	010	5
W	23	3	011	8
A	1	4	1000	3
D	4	4	1001	3
E	5	4	1010	4
N	14	4	1011	5
S	19	4	1100	5
U	21	4	1101	8
C	3	5	11100	4
L	12	5	11101	8
F	6	6	111100	4
H	8	6	111101	4
I	9	6	111110	5
T	20	6	111111	8

Таблица 3.3.3

Подсчет числа бит на передачу дерева регулярного кода

Ярус	Общее число вершин	Число концевых вершин n_i	Диапазон возможных 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
Всего				19

Именно так упорядочены буквы на рис. 3.3.2. и в таблице 3.3.3. В нашем примере после того как стала известна структура дерева, декодер предает сначала 8 битами номер буквы, имеющей длину слова 2. Затем нужно указать, какие 2 буквы из 255 имеют длину слова 3. Эти 2 буквы можно выбрать $\binom{255}{2}$ способами, передача номера комбинации потребует $\left\lceil \log\binom{255}{2} \right\rceil$. Продолжая, получим

$$\left\lceil \log\binom{256}{1} \right\rceil + \left\lceil \log\binom{255}{2} \right\rceil + \left\lceil \log\binom{253}{6} \right\rceil + \left\lceil \log\binom{247}{2} \right\rceil + \left\lceil \log\binom{245}{4} \right\rceil = 105 .$$

Итак, уточненный расчет числа бит дает для данного примера

$$l = 178 + 19 + 105 = 302 \text{ бит.} \qquad (3.3.2)$$

Второй способ кодирования эффективнее, но сложнее. Проблема состоит в нумерации комбинаций букв. Относительно простое решение этой задачи мы приведем ниже при изучении нумерационного кодирования.

Перейдем к получению оценки скорости универсального кода, работающего по описанному выше алгоритму. Из рассмотрения примера вытекает следующее утверждение.

Лемма 3.3.1. Полное кодовое дерево, имеющее M концевых вершин, имеет $M-1$ промежуточных вершин. Для полного описания дерева достаточно $2M-1$ бит.

Доказательство. Первое из двух утверждений леммы легко доказать с помощью метода математической индукции. Мы предоставляем это сделать читателю качестве элементарного упражнения. Для доказательства второго утверждения достаточно предложить алгоритм построения описания дерева с помощью двоичной последовательности длины $2M-1$. Такой алгоритм был описан выше в рамках примера 3.3.1. ■

Оценку асимптотической эффективности двухпроходного кодирования сформулируем в виде следующей теоремы.

Теорема 3.3.1. При двухпроходном кодировании с использованием кода Хаффмена дискретного постоянного источника с объемом алфавита M и энтропией H средняя скорость кодирования удовлетворяет неравенству

$$\overline{R} \le H + 1 + \frac{1}{n}\left(M \log M + 3M - 1\right). \qquad (3.3.3)$$

Доказательство. Для заданной последовательности $\mathbf{x}$ через $l_1(\mathbf{x})$ и $l_2(\mathbf{x})$ по-прежнему обозначаем длину первой и второй части кодового слова. Информация о коде может быть разбита на две части: информация о кодовом дереве и информация о том, какая из букв соответствует каждой вершине кодового дерева. Длина описания кодового дерева может быть оценена с использованием леммы 3.3.1. Указание буквы для одной вершины потребует не более $\lceil \log M \rceil$ бит. В результате имеем оценку

$$l_1(\mathbf{x}) \le 2M - 1 + M\lceil \log M \rceil \le M \log M + 3M - 1. \qquad (3.3.4)$$

Длина второй половины кодового слова подсчитывается как

$$l_2(\mathbf{x}) = \sum_{i=1}^{n} l(x_i) = \sum_{x \in X} \tau_n(x) l(x) = n \sum_{x \in X} \frac{\tau_n(x)}{n} l(x) =$$
$$= n \sum_{x \in X} \hat{p}_n(x) l(x) = n\mathbf{M}\left[l(x)\right] \le n(H(\hat{\mathbf{p}}_n) + 1). \qquad (3.3.5)$$

Поясним эти выкладки. Через $l(x)$ мы обозначили длину кодового слова для буквы x в построенном по $\mathbf{x}$ коде Хаффмена. В первом равенстве мы записали длину всей последовательности как сумму длин отдельных кодовых слов. Во втором равенстве мы сгруппировали слагаемые, соответствующие одинаковым буквам алфавита. Затем мы ввели обозначение $\hat{p}_n(x) = \tau_n(x)/n$ для оценок вероятностей букв по последовательности $\mathbf{x}$. Далее мы заменили сумму на математическое ожидание длины кодовых слов по распределению вероятностей $\hat{\mathbf{p}}_n = \{\hat{p}_n(x)\}$. После этого введено обозначение $H(\hat{\mathbf{p}}_n)$ для энтропии ансамбля с распределением вероятностей, задаваемым стохастическим вектором вероятностей $\hat{\mathbf{p}}_n$. Последнее неравенство основано на том, что средняя длина кодовых слов кода Хаффмена для источника с энтропией H не превышает $H+1$.

С учетом (3.3.4) и (3.3.5) средняя скорость кодирования для заданной последовательности $\mathbf{x}$ удовлетворяет неравенству

$$\overline{R}(\mathbf{x}) = \frac{l(\mathbf{x})}{n} = \frac{l_1(\mathbf{x}) + l_2(\mathbf{x})}{n} \le H(\hat{\mathbf{p}}_n) + 1 + \frac{1}{n}\left(M \log M + 3M - 1\right). \qquad (3.3.6)$$

Следующий шаг доказательства – усреднение по всем последовательностям $\mathbf{x}$. В правой части только энтропия зависит от $\mathbf{x}$. Поэтому рассмотрим ее отдельно.

$$\mathbf{M}\left[H(\hat{\mathbf{p}})\right] \le H\left(\mathbf{M}\left[\hat{\mathbf{p}}\right]\right) = H\,. \tag{3.3.7}$$

Здесь неравенство следует из выпуклости $\cap$ энтропии как функции распределения вероятностей. Для обоснования следующего перехода нужно убедиться в том, что

$$M\left[\hat{\mathbf{p}}_n\right] = \mathbf{p}\,, \tag{3.3.8}$$

где через $\mathbf{p}$ обозначен вектор, компонентами которого являются вероятности букв источника. Иными словами, нужно доказать, что в последнем векторном тождестве равенство имеет место для каждой компоненты, т.е.

$$\mathbf{M}\left[\frac{\tau_n(a)}{n}\right] = p(a), \quad a \in X\,.$$

Чтобы это доказать, введем индикаторную функцию

$$\chi_a(x) = \begin{cases} 1, & \text{при } x = a, \\ 0, & \text{при } x \ne a. \end{cases}$$

Заметим, что

$$\mathbf{M}\left[\chi_{\mathbf{a}}(x)\right] = 1 \times p(a) + 0 \times (1 - p(a)) = p(a).$$

Теперь можно записать

$$\mathbf{M}\left[\frac{\tau_n(a)}{n}\right] = \frac{1}{n}\mathbf{M}\left[\sum_{i=1}^{n}\chi_a(x_i)\right] = \frac{1}{n}\sum_{i=1}^{n}\mathbf{M}\left[\chi_a(x_i)\right] = p(a), \quad a \in X\,.$$

Отсюда следует (3.3.8). В свою очередь, используя (3.3.8), в результате усреднения в левой и правой частях (3.3.7), мы приходим к доказываемому результату (3.3.4). ■

Обсудим полученную оценку эффективности кодирования. Избыточность кодирования удовлетворяет неравенству вида

$$r = \overline{R} - H \le 1 + \frac{K}{n}, \tag{3.3.9}$$

где первое слагаемое в правой части обусловлено избыточностью использованного побуквенного кода, второе – отсутствием информации о статистике источника. При $n \to \infty$ влияние второго слагаемого становится пренебрежимо малым и эффективность двухпроходного кодирования стремится к эффективности кодирования при известном распределении вероятностей на буквах источника.

Асимптотический результат легко распространить на случай источников с памятью. Для этого описанный способ кодирования нужно применить к блокам достаточно большой длины. Большого практического значения такой подход не имеет, поскольку с увеличением длины блоков экспоненциально растет объем алфавита и, соответственно, длина первой (служебной) части кодового слова.

3.4. Двухпроходное арифметическое кодирование

Посмотрим, насколько повысится эффективность кодирования, если заменить побуквенный код арифметическим. Мы помним, что при известной статистике источника вместо избыточности на сообщение порядка 1 бита мы получаем избыточность порядка $2/n$. При неизвестной статистике результат может оказаться другим, поскольку информация, необходимая для описания арифметического кода, может потребовать больших затрат, чем информация о коде Хаффмена.

Начнем с примера.

Пример 3.4.1. Снова рассмотрим пословицу

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

Арифметический код однозначно определяется вероятностями букв, которые, в свою очередь, могут быть вычислены по композиции последовательности. Вся необходимая

информация представлена в таблице 3.4.1.

Таблица 3.4.1

Подсчет длины кодового слова при арифметическом кодировании для примера 3.4.1.

Буква	Композиция	Затраты на передачу буквы
I	1	-log(1/50)
F	1	-log(1/50)
_	12	-log(12/50)
W	5	-log(5/50)
E	4	-log(4/50)
C	2	-log(2/50)
A	4	-log(4/50)
N	3	-log(3/50)
O	5	-log(5/50)
T	1	-log(1/50)
D	4	-log(4/50)
S	3	-log(3/50)
U	2	-log(2/50)
L	2	-log(2/50)
H	1	-log(1/50)

Подсчет затрат на кодирование последовательности при известном коде дает результат

$$l_2 = \lceil 176{,}44 \rceil + 1 = 178 \text{ бит.}$$

Сравнивая с предыдущим примером, убеждаемся, что это число в точности совпало с затратами при кодировании кодом Хаффмена.

К этому числу нужно добавить затраты на передачу информации о композиции. В следующем параграфе при анализе нумерационного кодирования мы исследуем различные способы передачи информации о композиции. Здесь мы приводим готовый результат

$$l_1 = 133 \text{ бит}$$

Суммарная длина кодового слова

$$l = l_1 + l_2 = 311 \text{ бит.} \quad \blacksquare$$

Обобщение этого способа кодирования на произвольные последовательности сформулируем в виде теоремы.

Теорема 3.4.1. При двухпроходном арифметическом кодировании дискретного постоянного источника с объемом алфавита M и энтропией H средняя скорость кодирования удовлетворяет неравенству

$$\overline{R} \le H + \frac{M \log(n+1) + M + 1}{n}. \qquad (3.4.1)$$

Доказательство. Сначала рассмотрим затраты на передачу композиции. Композиция представляет собой вектор длины M, в котором каждая из компонент принимает одно из не более чем $(n+1)$ значений ($\tau_n(a) \in \{0,...,n\}$). Это означает, что на передачу одной компоненты достаточно $\log(n+1)$ бит. Поскольку сумма компонент в точности равна n, последняя компонента вектора может быть вычислена по предыдущим, следовательно, достаточно передать $M-1$ компоненту композиции. Таким образом,

$$l_1(\mathbf{x}) \le (M-1)\lceil \log(n+1) \rceil \le M \log(n+1) + M - 1.$$

Для второй части кодового слова затраты составят

$$l_2(\mathbf{x}) = \left\lceil -\log \frac{\hat{p}(\mathbf{x})}{2} \right\rceil \le -\log \hat{p}(\mathbf{x}) + 2 =$$

$$= -\log \prod_{i=1}^{n} \hat{p}_n(x_i) + 2 = -\sum_{i=1}^{n} \log \hat{p}_n(x_i) + 2 =$$

$$= -\sum_{x \in X} \tau(x) \log \hat{p}_n(x) + 2 = n\left(-\sum_{x \in X} \hat{p}_n(x) \log \hat{p}_n(x) \right) + 2 =$$

$$= nH(\hat{\mathbf{p}}_n) + 2.$$

Здесь через $\hat{p}(\mathbf{x})$ обозначена оценка вероятности последовательности, вычисленная по оценкам вероятностей букв. Первый переход основан на формуле для длины кодового слова при арифметическом кодировании. Далее мы использовали неравенство $\lceil a \rceil \le a + 1$, затем – независимость букв источника. После этого изменен порядок суммирования и в конце использована формула для энтропии ансамбля.

Суммируя затраты, получаем среднюю скорость кодирования

$$\overline{R}(\mathbf{x}) \le H(\hat{\mathbf{p}}) + \frac{M \log(n+1) + M + 1}{n}.$$

Для завершения доказательства нужно усреднить по всем последовательностям $\mathbf{x}$ и воспользоваться выпуклостью энтропии, как это сделано при доказательстве теоремы 3.3.1. ■

Из теоремы следует, что при $n \to \infty$ скорость универсального двухпроходного кодирования стремится к энтропии источника. Мы получили этот результат для источника без памяти. Его легко обобщить на источники с памятью. При этом нет необходимости рассматривать кодирование блоков. Можно вместо одномерного распределения передавать условные вероятности букв и использовать эти условные вероятности при арифметическом кодировании.

Из теоремы 3.4.1 следует асимптотическая оценка избыточности для двухпроходного арифметического кодирования

$$r = \overline{R} - H \le \frac{M \log n + K}{n},$$

где K - константа, не зависящая от n.

3.5. Нумерационное кодирование

Сейчас мы рассмотрим один из исторически первых методов универсального кодирования. Его автором является Фитингоф Б. М.

Считаем, что алфавитом источника служит множество чисел $X = \{0,...,M-1\}$. Пусть $\mathbf{x} = (x_1,...,x_n)$ – последовательность на выходе источника. Рассмотрим следующий способ двухпроходного кодирования.

Кодовое слово нумерационного кода состоит из двух частей. Первая часть содержит закодированный равномерным кодом номер композиции $\boldsymbol{\tau}_n(\mathbf{x}) = (\tau_n(0),...,\tau_n(M-1))$ в списке всех возможных композиций последовательностей длины n над алфавитом X. Вторая часть содержит закодированный равномерным кодом номер данной последовательности $\mathbf{x}$ в лексикографически упорядоченном списке последовательностей с одинаковой композицией $\boldsymbol{\tau}_n(\mathbf{x})$.

В параграфе 3.2 подсчитано число различных композиций (формула (3.2.7))

$$N_\tau(n,M)=\binom{n+M-1}{M-1} \tag{3.5.1}$$

и число последовательностей с фиксированной композицией $N(\boldsymbol{\tau})$ (формула (3.2.6))

$$N(\boldsymbol{\tau})=\frac{n!}{\tau_0!\ldots\tau_{M-1}!}. \tag{3.5.2}$$

Для заданной последовательности $\mathbf{x}$ через $l_1(\mathbf{x})$ и $l_2(\mathbf{x})$ обозначим длину первой и второй частей кодового слова Из формул (3.5.1) и (3.5.2) получаем формулу для длины $l(\mathbf{x})$ кодового слова:

$$l(\mathbf{x})=l_1(\mathbf{x})+l_2(\mathbf{x})=\lceil\log N_\tau(M)\rceil+\lceil\log N(\boldsymbol{\tau})\rceil=\left\lceil\log\binom{n+M-1}{M-1}\right\rceil+\left\lceil\log\frac{n!}{\prod_{i=0}^{M-1}\tau_i(\mathbf{x})!}\right\rceil. \tag{3.5.3}$$

Пример 3.5.1. Продолжим эксперименты с пословицей

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

Непосредственный подсчет по формулам (3.5.1) – (3.5.3) приводит к результатам

$$l_1=\left\lceil\log\binom{50+255}{255}\right\rceil=190 \text{ бит},$$

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

Сравнивая с предыдущими примерами, видим, что l_1 неожиданно велико, а l_2 рекордно мало. Не может быть, чтобы не существовало более эффективного правила передачи композиции.

С другой стороны, формула (3.5.1) точна и, следовательно, уменьшение бит может быть получено только в том случае, если мы откажемся от равномерного кодирования номеров композиций. Для каких-то композиций мы должны тратить еще больше бит, чем раньше, и тогда, возможно, для некоторых композиций, кодирование станет значительно более эффективным. По-видимому, в «привелегированое» положение должны быть поставлены те композиции, для которых «сжатие» в принципе осуществимо, т.е. композиции, содержащие сильно разнящиеся величины.

Мы будем отдельно передавать композицию, а затем – указывать, какие буквы соответствуют различным компонентам композиции. Композицию перед передачей отсортируем. В данном примере сортированная композиция имеет вид $\tilde{\boldsymbol{\tau}}=(\tilde{\tau}_0,\ldots,\tilde{\tau}_{M-1})$=(12, 5, 5, 4, 4, 4, 3, 3, 2, 2, 2, 1, 1, 1, 1, 0,…., 0). Для элементов композиции имеют место неравенства

$$1\le\tilde{\tau}_0\le n,$$

$$1\le\tilde{\tau}_j\le\tilde{\tau}_{j-1},\quad j>0,\ \tau_j>0.$$

Исходя из этих соотношений, находим, что число различных упорядоченных композиций не превышает величины

$$n\prod_{j:\tau_j>0}\tau_j. \tag{3.5.4}$$

Это соотношение следует из (3.2.1). Осталось установить соответствие между буквами и компонентами композиции. Для одинаковых значений композиции порядок следования букв несущественен. Введем композицию сортированной композиции и обозначим ее $\boldsymbol{\tau}'=(\tau'_0,\tau'_1,\ldots)$. В данном примере $\boldsymbol{\tau}'$=(1,2,3,2,3,4,241). Достаточно передать, в какую из этих 7 категорий попала каждая буква алфавита. Для подсчета числа различных вариантов годится формула (3.2.6). Поэтому затраты на передачу композиции можно подсчитать как

$$l_1 = \left\lceil \log\left(n \prod_{j:\tau_j>0} \tau_j \right) \right\rceil + \left\lceil \log\left(\frac{M!}{\prod_j \tau_j'!} \right) \right\rceil = 25 + 108 = 133 \text{ бит}. \qquad (3.5.5)$$

Окончательный результат для нумерационного кодирования составляет

$$l = l_1 + l_2 = 283 \text{ бит} \qquad \blacksquare \qquad (3.5.6)$$

Оценим теперь эффективность нумерационного кодирования для произвольного дискретного постоянного источника с неизвестным распределением вероятностей. Нас интересует случай когда длина последовательности n достаточно велика и выполняется неравенство $n >> M$. Для оценки числа композиций вместо формулы (3.5.1) мы воспользуемся оценкой

$$N_\tau(n, M) \le (n+1)^{M-1}. \qquad (3.5.7)$$

Она вытекает из того, что каждая из компонент принимает одно из $(n+1)$ значений от 0 до n и последняя компонента композиции однозначно вычисляется по первым $M-1$. Для оценки числа последовательностей с заданной композицией воспользуемся (3.2.10). Получим

$$l(\mathbf{x}) = l_1(\mathbf{x}) + l_2(\mathbf{x}) = \lceil \log N_\tau(M) \rceil + \lceil \log N(\boldsymbol{\tau}) \rceil \le$$

$$\le nH(\hat{\mathbf{p}}) - \frac{M-1}{2}\log(2\pi n) - \frac{1}{2}\sum_i \log(\hat{p}_i) + (M-1)\log(n+1) + 1.$$

Теперь поделим обе части на n и усредним по всем последовательностям $\mathbf{x}$. Результат сформулируем в виде теоремы.

Теорема 3.5.1. При нумерационном кодировании дискретного постоянного источника с объемом алфавита M и энтропией H средняя скорость кодирования удовлетворяет неравенству

$$\overline{R} \le H + \frac{M-1}{2}\frac{\log(n+1) + K}{n}, \qquad (3.5.8)$$

где величина K не зависит от длины последовательности n.

Сравнивая результат с оценкой (3.4.1) средней скорости двухпроходного арифметического кодирования убеждаемся, что при больших n избыточность нумерационного кодирования примерно в 2 раза меньше.

3.6. Реализация нумерационного кодирования

Рассматривая методы двухпроходного кодирования, мы убедились в эффективности нумерационного кодирования. Этот метод включает два этапа кодирования и каждый этап требует передачи номера заданной последовательности в некотором списке. При большой длине кодируемой последовательности списки могут быть большими и вычисление номера последовательности может быть слишком сложным.

Вернемся к примеру 3.5.1. Длина последовательности букв источника была небольшой, $n = 50$. В то же время длины кодовых последовательностей для первой и второй части кодового слова получились равными 133 и 150 бит. Это значит, что списки содержали порядка 2^{133} и 2^{150} последовательностей. Понятно, что табличные или переборные методы для кодирования и декодирования неприемлемы.

В этом параграфе мы покажем, что нумерационное кодирование может быть выполнено с полиномиальной по n сложностью. Сначала мы опишем исторически первый алгоритм, предложенный Бабкиным В. Ф. [Bab71]. Затем рассмотрим весьма практичный способ нумерации последовательностей, основанный на арифметическом кодировании.

Начнем с нумерации двоичных последовательностей. Рассмотрим множество

двоичных последовательностей длины n с числом единиц (весом Хэмминга) w. Это множество, образующее сферу радиуса w в n – мерном хэммингове пространстве, будем обозначать $S_n(w)$. Как мы знаем, общее число таких последовательностей равно $|S_n(w)| = \binom{n}{w}$. Задача кодирования состоит в том, чтобы для произвольной последовательности $\mathbf{x} \in S_n(w)$ вычислить ее номер $J(\mathbf{x})$ в лексикографически упорядоченном множестве $S_n(w)$. Для декодирования нужно уметь по номеру последовательности $J(\mathbf{x})$ вычислить саму последовательность $\mathbf{x}$.

Обозначим через $p_1,...,p_w$ номера ненулевых позиций в $\mathbf{x}$, вычисленные от последнего символа. Например для $\mathbf{x} = (0100001010)$ номера позиций равны $p_1 = 2$, $p_2 = 4$, $p_3 = 9$. Наряду с $J(\mathbf{x})$ будем для номера той же последовательности использовать обозначение $J(p_1,...,p_w)$.

Теорема 3.6.1 Номер последовательности с номерами ненулевых позиций $p_1,...,p_w$ равен

$$J(p_1,...,p_w) = \binom{p_w - 1}{w} + \binom{p_{w-1} - 1}{w-1} + ... + \binom{p_1 - 1}{1}. \qquad (3.6.1)$$

Доказательство. Подсчитаем, сколько последовательностей лексикографически предшествует последовательности, содержащей старшую единицу на позиции p_w. Это прежде всего все те последовательности, которые имеют все w единиц на позициях с номерами (вычисленными от конца) 1, 2,…, $p_w - 1$. Число таких последовательностей равно $\binom{p_w - 1}{w}$. Кроме того, нужно учесть все те последовательности, которые содержат единицу на позиции с номером p_w, но их остальные $w-1$ единиц расположены таким образом, что они предшествуют последовательности с единицами на позициях $p_1,...,p_{w-1}$. Согласно нашим обозначениям число таких последовательностей равно $J(p_1,...,p_{w-1})$. Итак,

$$J(p_1,...,p_w) = \binom{p_w - 1}{w} + J(p_1,...,p_{w-1}). \qquad (3.6.2)$$

Поясним эту формулу примером. Последовательности $\mathbf{x} = (0100001010)$ предшествуют, во-первых, все последовательности веса 3, содержащие единицы только на последних 8 позициях (их число $\binom{8}{3}$), во-вторых, последовательности вида (01….) такие, что подпоследовательность из последующих 8 символов младше рассматриваемой (таких последовательностей $J(2,4)$. Таким образом, в нашем случае $J(2,4,9) = \binom{8}{3} + J(2,4)$.

Формулу (3.6.2) можно применить ко второму слагаемому в (3.6.2). Проделав это $w-1$ раз, получим (3.6.1). ■

Для $\mathbf{x} = (0100001010)$ имеем $J(2,4,9) = \binom{8}{3} + \binom{3}{2}. + \binom{1}{1}$=60. Для передачи этого числа потребуется 7 бит, поскольку $\left\lceil \log\binom{10}{3} \right\rceil = \lceil \log 120 \rceil = 7$.

Рассмотрим задачу декодирования, т.е. вычисления $\mathbf{x}$ по $J(\mathbf{x})$. В решении задачи нам поможет тождество (3.2.13). Перепишем его в виде

$$\binom{n-1}{w}=\binom{n-2}{w}+\binom{n-3}{w-1}+\ldots+\binom{n-w+1}{1}+1. \qquad (3.6.3)$$

Из него следует, что единица на позиции с номером n вносит в сумму $J(\mathbf{x})$ слагаемое, величина которого по меньшей мере на 1 больше, чем могла бы внести любая комбинация из w единиц на позициях с меньшими (от конца) номерами. Поэтому, сравнивая $J(\mathbf{x})$ с $\binom{n-1}{w}$, мы можем вынести однозначное решение о том, имеется единица на позиции с номером n или нет. Приходим к следующему алгоритму.

Алгоритм декодирования нумерационного кода.

1. $S=J(\mathbf{x})$, $W=w$, $N=n$.

2. До тех пор пока $N>0$ выполняются вычисления:

2.1. Если $\binom{N-1}{W}\le S$ и $W>0$, то $x_N=1$, $S\leftarrow S-\binom{N-1}{W}$, $W\leftarrow W-1$, переходим к 2.3, в противном случае $x_N=0$.

2.2. Полагаем $N\leftarrow N-1$.

В таблице 3.6.1 показаны вычисления, выполняемые декодером нумерационного кода последовательности длины $n=10$ веса $w=3$ при поступлении на его вход числа $J=60$.

Таблица 3.6.1

Пример декодирования нумерационного кода

Номер текущей позиции N	Число единиц в оставшихся позициях W	Текущее значение суммы S	$\binom{N-1}{W}$	Принятое решение x_n
10	3	60	72	0
9	3	60	56	1
8	2	4	21	0
7	2	4	15	0
6	2	4	10	0
5	2	4	6	0
4	2	4	3	1
3	1	1	2	0
2	1	1	1	1
1	0	0	0	0

В процессе кодирования и декодирования нумерационного кода многократно используются значения комбинаторных коэффициентов. При небольших длинах последовательностей биномиальные коэффициенты можно вычислить заранее и хранить в памяти кодера и декодера. Для вычисления коэффициентов при больших длинах n можно воспользоваться рекуррентной формулой (3.2.12). Эта формула позволяет избежать умножений. В некоторых случаях удобнее бывает другая рекуррентная формула

$$\binom{n}{w}=\frac{n-w+1}{w}\binom{n}{w-1}.$$

Нужно иметь при этом ввиду, что все вычисления должна выполняться без округлений, т.е. разрядность чисел, над которыми выполняются арифметические операции должна быть равной длине кодовой последовательности.

Мы закончили рассмотрение нумерации двоичных последовательностей. Нумерация последовательностей над произвольным алфавитом может быть выполнена с помощью

нумерации двоичных последовательностей следующим очевидным способом. Сначала передаются номера позиций первого из символов алфавита, затем для остальных позиций – номера позиций второго символа алфавита и т.д.

Другой подход к реализации нумерационного кодирования основан на использовании арифметического кодирования. Этот метод мы сразу рассмотрим применительно к последовательностям над произвольным алфавитом $X = \{0,...,M-1\}$. Пусть $\boldsymbol{\tau} = (\tau_0,...,\tau_{M-1})$ обозначает композицию вектора. Задача нумерационного кодирования, строго говоря, состоит в вычислении номера некоторой последовательности $\mathbf{x} = (x_1,...,x_n)$ в лексикографически упорядоченном списке всех последовательностей с заданной композицией $\boldsymbol{\tau}$. Эквивалентная с практической точки зрения задача состоит в том, чтобы выполнить взаимно-однозначное отображение множества последовательностей X^n на множество двоичных последовательностей длины

$$l = \lceil \log N(\boldsymbol{\tau}) \rceil = \left\lceil \log \frac{n!}{\tau_0!...\tau_{M-1}!} \right\rceil. \qquad (3.6.4)$$

Следующий алгоритм решает эту задачу.

Алгоритм арифметического нумерационного кодирования.

Заданы композиция $\boldsymbol{\tau} = (\tau_0,...,\tau_{M-1})$ и последовательность $\mathbf{x} = (x_1,...,x_n)$ с композицией $\boldsymbol{\tau}$.

1. Инициализация. Вычисляем вероятности букв $p_j = \tau_j / n$, и кумулятивные вероятности $q_j = \sum_{k=0}^{j-1} p_k$, $j = 0,...,M-1$. Полагаем $F = 0$, $G = 1$.
2. Основной цикл. Для $i = 1,...,n$ выполняем следующие вычисления:
   - $F \leftarrow F + q(x_i) \times G$;
   - $G \leftarrow p(x_i) \times G$;
   - Модифицируем композицию: $\tau_{x_i} \leftarrow \tau_{x_i} - 1$;
   - $n \leftarrow n - 1$;
   - Вычисляем $p_j = \tau_j / n$, $q_j = \sum_{k=0}^{j-1} p_k$, $j = 0,...,M-1$.

3. Кодовое слово для $\mathbf{x}$ формируется как первые $l = \lceil \log G \rceil$ разрядов после запятой в двоичном представлении числа F. ■

Заметим, что в алгоритме арифметического кодирования к числу F на последнем шаге кодирования прибавлялось число $G/2$ и длина кодового слова увеличивалась на 1, что соответствовало применению кода Гилберта-Мура. Напомним, что необходимость этих шагов была обусловлена тем, что мы не могли гарантировать, что лексикографически упорядоченные последовательности упорядочены по убыванию вероятностей. При нумерационном кодировании последовательностей с одинаковой композицией мы всем таким последовательностям приписываем одинаковые вероятности. Это дает возможность отказаться от смещения вероятностей и связанного с ним увеличения на 1 длины кодовой последовательности, т.е. можно применять код Шеннона.

Нетрудно убедиться в том, что имеет место равенство

$$G = \left(\frac{n!}{\tau_0!...\tau_{M-1}!} \right)^{-1},$$

следовательно, представленный алгоритм формирует кодовое слово минимальной возможной длины.

Декодером для данного кода будет арифметический декодер, который, заранее зная композицию последовательности, обновляет вероятности и кумулятивные вероятности

букв точно так же, как это делалось при кодировании.

3.7. Адаптивное кодирование

Мы переходим к изучению кодирования без задержки. Кодеру при поступлении на его вход очередного сообщения теперь недоступна информация о сообщениях, которые появятся в будущем. Поэтому способ кодирования текущего сообщения будет зависеть только от того, какими были предыдущие сообщения.

Естественным решением задачи является следующий подход. Кодер (и декодер) по последовательности уже переданных сообщений оценивают вероятности возможных значений следующего сообщения и строят для него код в соответствии с этими оценками вероятностей. В рамках этой общей схемы есть несколько степеней свободы выбора конкретной модификации алгоритма. В частности, можно менять следующие характеристики алгоритма

- длину последовательности сообщений, по которой вычисляются оценки вероятностей;
- формулы для вычисления оценок вероятностей;
- способ кодирования при известных оценках вероятностей.

Обсудим кратко эти возможности. Начнем с вопроса о выборе длины «окна», т.е. длины последовательности, по которой оцениваются вероятности сообщений. Интуитивно ясно, что увеличение длины окна должно приводить к повышению точности оценок, и, как следствие, к повышению эффективности кодирования. Это соображение верно, если верна гипотеза о стационарности источника. В противном случае выбор короткого окна может оказаться более удачным, поскольку позволит кодеру и декодеру быстрее адаптироваться к изменениям статистических свойств источника. Более детальный анализ показывает, что и для стационарного источника достаточно выбирать окно с длиной в несколько раз превышающей объем алфавита источника. Дальнейшее увеличение длины окна не приведет заметному уменьшению избыточности. Исходя из этого, кажется целесообразным всегда выбирать окна конечной длины. Однако на пути реализации такого подхода имеется «подводный камень»: он имеет более высокую сложность. Дело в том, что при «бесконечном окне» (когда окном служит вся предшествующая последовательность сообщений) вся необходимая информация о предшествующих сообщениях хранится в виде счетчиков числа появления сообщений (число счетчиков равно объему алфавита источника). При окне конечной длины при поступлении от источника нового сообщения нужно не только нарастить на 1 значения соответствующего счетчика, но и уменьшить на 1 значение счетчика для той буквы, которая в данный момент времени оказалась за пределами окна. Для этого придется помнить всю последовательность букв, хранящихся в окне. С учетом перечисленных факторов обычно вероятности оцениваются по всей предшествующей последовательности.

Выбор формул для оценивания вероятностей и выбор способа кодирования оказываются связанными. Казалось бы, оценка вероятности того, что $x_{n+1} = a$, должна вычисляться по последовательности $x_1,...,x_n$ по формуле

$$\hat{p}_n(a) = \frac{\tau_n(a)}{n}, \tag{3.7.1}$$

где через $\tau_n(a)$ обозначено число появлений буквы a в последовательности длины n. Для некоторых сообщений оценка вероятности окажется равной нулю. Это не вызовет серьезных проблем, если для кодирования используется, например, код Хаффмена. Однако при использовании кодов Шеннона, Гилберта-Мура или арифметического кодирования нулевые значения вероятностей недопустимы.

В литературе по кодированию источников можно найти много способов записи оценок вероятностей букв (см., например, [Kr89]). Например, можно воспользоваться формулой

$$\hat{p}_n(a) = \frac{\tau_n(a)+1}{n+M}. \qquad (3.7.2)$$

Смысл этого приближения состоит в том, что в числителе формулы добавлена 1 к счетчику числа появлений букв и тем самым мы исключили нулевые вероятности. К знаменателю пришлось добавить объем алфавита, иначе нарушилось бы условие нормировки вероятностей. При увеличении длины n различие между формулами (3.7.1) и (3.7.2) становится несущественным, их можно считать асимптотически эквивалентными. Однако, при длине n сопоставимой с объемом алфавита M оценки, подсчитанные по формуле (3.7.2) будут значительно меньше несмещенных оценок (3.7.1), и поэтому затраты на кодирование букв (минус логарифмы вероятностей букв) будут заметно больше. Отсюда следует, что, по крайней мере для коротких последовательностей, формула (3.7.2) неэффективна.

Более практичен подход, основанный на введение в алфавит дополнительной буквы, которую мы назовем *esc*-символом, что, как мы увидим позже, вполне соответствует способу ее применения.

Тем буквам алфавита, которые уже встречались в последовательности из n предшествующих букв, приписываются вероятности

$$\hat{p}_n(a) = \frac{\tau_n(a)}{n+1}, \quad \tau_n(a) > 0.$$

Если же в момент времени $n+1$ появилась буква, которой ранее ни разу не было, то передается *esc*-символ, которому приписывается вероятность

$$\hat{p}_n(esc) = \frac{1}{n+1},$$

а затем передается сама буква, при этом всем буквам, которые не встречались, приписываются одинаковые вероятности.

Эту стратегию кодирования (назовем ее А-алгоритмом) можно подытожить в виде следующей формулы

$$\hat{p}_n(a) = \begin{cases} \dfrac{\tau_n(a)}{n+1}, & \text{если } \tau_n(a) > 0; \\ \dfrac{1}{n+1}\dfrac{1}{M-M_n}, & \text{если } \tau_n(a) = 0, \end{cases} \qquad (3.7.3)$$

где M_n обозначает число различных букв, содержащихся в последовательности длины n.

Подход, основанный на использовании *esc*-символа, мы исследуем далее на примере, а затем оценим его эффективность применительно к произвольному дискретному постоянному источнику.

Пример 3.7.1. (Алгоритм А) Снова кодируем последовательность

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

Чтобы определить длину кодового слова, нужно вычислить число G, представляющее собой оценку вероятности последовательности. До начала кодирования $G = 1$. После кодирования буквы «I» в соответствии с (3.7.3) $G = 1/256$. После второго и третьего шагов имеем

$$G = 1 \cdot \frac{1}{2} \cdot \frac{1}{256} \cdot \frac{1}{3} \cdot \frac{1}{255}$$

и т.д.. Заметим, что следующему появлению пробела будет соответствовать сомножитель $1/6$, третьему пробелу – сомножитель 2/13,…, последнему – 11/47. Аналогично учтем вклад каждой буквы и получим

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

Следовательно, длина кодового слова равна

$$l = \lceil -\log G \rceil + 1 = 291 \text{ бит.}$$

Результат получился почти такой же, как при нумерационном кодировании, хотя данный алгоритм является однопроходным. Таким образом, возможность «заглянуть в будущее» в данном примере не дает большого выигрыша в кодировании. ■

Вычисления, выполненные в последнем примере, легко обобщаются на случай произвольной последовательности $\mathbf{x} = (x_1,...,x_n)$ с композицией $(\tau_1,...,\tau_M)$. Получим

$$G = \frac{\prod_{i=1}^{M_n}(\tau_i - 1)!}{n!} \cdot \frac{(M - M_n)!}{M!} = \frac{\prod_{i=1}^{M_n}\tau_i!}{n!} \cdot \frac{(M - M_n)!}{M!\prod_{i=1}^{M_n}\tau_i} \geq \frac{\prod_{i=1}^{M_n}\tau_i!}{n!} M^{-M}(n+1)^{-M}. \qquad (3.7.4)$$

Применим асимптотические оценки из раздела 3.2. Повторив выкладки, использованные при выводе (3.5.8), получим следующий результат.

Теорема 3.7.1. При адаптивном арифметическом кодировании дискретного постоянного источника с объемом алфавита M и энтропией H средняя скорость кодирования удовлетворяет неравенству

$$\overline{R} \leq H + \frac{M}{2}\frac{\log(n+1) + K}{n}, \qquad (3.7.5)$$

где величина K не зависит от длины последовательности n.

Заметим, что асимптотические оценки эффективности адаптивного кодирования получаются примерно такими же, что и при нумерационном кодировании. Это вполне согласуется с цифрами, полученными в примере 3.7.1 для короткой последовательности на выходе источника.

Остановимся на практических аспектах адаптивного кодирования.

Для некоторых приложений использование арифметического кодирования может оказаться нецелесообразным. В этом случае оно может быть заменено, например, кодированием кодом Хаффмена. При этом, поскольку на каждом шаге кодирования распределение вероятностей букв меняется, возникает необходимость построения нового кода Хаффмена после кодирования каждой буквы. На первый взгляд такой метод кажется неприемлемо сложным, но при ближайшем рассмотрении оказывается, что он может быть реализован достаточно эффективно. Эта задача успешно решена в работе Р. Галлагера [G78]. Поскольку за один шаг распределение вероятностей и, соответственно, код Хаффмена сильно измениться не могут, фактическое число операций, требуемое для модификации кода, оказывается весьма небольшим.

Отметим, что алгоритм А (формула (3.7.3)) – простой для понимания и анализа алгоритм, но не самый лучший. Действительно, например, после кодирования первой буквы источника на втором шаге мы имеем одинаковые вероятности *esc*-символа и единственного известного кодеру и декодеру первого символа. Это не совсем справедливо, поскольку с *esc*-символом ассоциируются все остальные буквы алфавита. Вероятно, кодирование станет более эффективным, если на первых шагах *esc*-символ будет иметь больший вес, но с течением времени приписываемая ему вероятность будет уменьшаться.

Рассмотрим альтернативную оценку

$$\hat{p}_n(a) = \begin{cases} \dfrac{\tau_n(a) - 1/2}{n}, & \text{если } \tau_n(a) > 0; \\ \dfrac{M_n}{2n}\dfrac{1}{M - M_n}, & \text{если } \tau_n(a) = 0. \end{cases} \qquad (3.7.6)$$

Кодирование с использованием этой оценки назовем D-алгоритмом. Легко убедиться, что условие нормировки выполняется и что по сравнению с А-алгоритмом заметно увеличена вероятность, приписываемая новым символам по крайней мере на первых шагах кодирования. Проверим эффективность D-метода на примере.

Пример 3.7.2. (Алгоритм D). Снова кодируем последовательность

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

Чтобы определить длину кодового слова, нужно вычислить число G. До начала кодирования $G=1$. После нескольких первых шагов имеем

$$G=1\cdot\frac{1}{2}\cdot\frac{1}{256}\cdot\frac{2}{4}\cdot\frac{1}{255}\cdot\frac{3}{6}\cdot\frac{1}{254}\frac{4}{8}\cdot\frac{1}{253}\frac{1}{10}\cdot\frac{5}{12}\cdot\frac{1}{252}\ldots$$

Заметим, что в знаменателе будем иметь произведение двух арифметических прогрессий: 2×4×…×100×256×255×…×242. В числителе *esc*-символы породят произведение 1×2×3×…×14, а каждая буква, появившаяся τ раз, вызовет появление произведения $1\times3\times5\times\ldots\times(2\tau-3)$. Будем использовать обозначения $(2n-1)!!=1\times3\times\ldots\times(2n-1)$, $(2n)!!=2\times4\times\ldots\times(2n)$. В целом для всей последовательности получим

$$G=\frac{(2\times12-3)!!((2\times5-3)!!)^2((2\times4-3)!!)^3((2\times3-3)!!)^2}{100!!}\cdot\frac{14!}{256\cdot255\cdot\ldots\cdot242}.$$

Следовательно, длина кодового слова равна

$$l=\lceil-\log G\rceil+1=283 \text{ бит.}$$

Результат получился лучше, чем кодировании по алгоритму A и в точности такой же, как и при нумерационном кодировании. Таким образом, снова убеждаемся, что возможность «заглянуть в будущее» не всегда дает выигрыш при кодировании. ■

В завершение параграфа отметим, что алгоритмы A и D совпадают с формулами для оценок условных вероятностей символов при заданном контексте, применяемыми в алгоритмах универсального кодирования PPMA[CW84] и PPMD[H93]. PPM алгоритмы будут рассмотрены в разделе 4.

3.9. Сравнение алгоритмов

Для удобства сравнения алгоритмов их характеристики сведены в таблицу 3.9.1. В этой таблице в формулах для асимптотической избыточности через n обозначена длина последовательности, M - объем алфавита, а величины K_i, $i=1,...,4$ представляют собой константы, независящие от n.

Полученные результаты показывают, что, если не принимать во внимание сложность алгоритмов, предпочтения заслуживают нумерационное кодирование и адаптивное арифметическое кодирование. Неожиданными кажется тот факт, что возможность выполнения двух проходов при кодировании почти не сказывается на достижимом сжатии. Причина в том, что однопроходное кодирование обеспечивает практически оптимальное кодирование (его избыточность блика к нижней границе), поэтому выигрыш в принципе большим быть не может.

Таблица 3.9.1

Сравнение алгоритмов универсального кодирования

Алгоритм	Число проходов	Асимптотическая избыточность (бит)	Длина кодового слова для рассмотренного примера
Двухпроходное кодирование с использованием кода Хаффмена	2	$1 + K_1 / n$	302
Двухпроходное арифметическое кодирование	2	$\frac{M \log n + K_2}{n}$	311
Нумерационное кодирование	2	$\frac{M \log n + K_3}{2n}$	283
Адаптивное арифметическое кодирование (алгоритм A)	1	$\frac{M \log n + K}{2n}$	291
Адаптивное арифметическое кодирование (алгоритм D)	1	$\frac{M \log n + K}{2n}$	283

Раздел 4
Алгоритмы кодирования источников, применяемые в архиваторах

В разделе 2 мы познакомились с методами кодирования при известной статистике источника. Было показано, что, применяя арифметическое кодирование, можно получить скорость кодирования сколь угодно близкую к нижнему пределу – к скорости создания информации источником. Далее в разделе 3 мы получили такой же оптимистический результат и для случая, когда статистические характеристики источника неизвестны. Понятно, что методы раздела 3 можно применять к последовательностям букв (вместо отдельных букв) и тем самым добиться скорости кодирования сколь угодно близкой к энтропии на сообщение. Другой очевидный способ развития этих методов – аппроксимация модели источника цепью Маркова достаточно высокого порядка. Оба пути оказываются непрактичными. Во-первых, их сложность быстро растет с увеличением длины блока или порядка аппроксимирующей цепи Маркова. Во-вторых, при кодировании последовательностей конечной длины, в частности, при кодировании типичных файлов пользователей персональных компьютеров, асимптотически оптимальные методы не всегда дают наилучший результат. Точнее говоря, различные способы кодирования, имея асимптотически эквивалентные характеристики, могут существенно различаться по эффективности при использовании в реальных условиях.

В данном разделе мы опишем наиболее эффективные способы универсального кодирования источников. Наиболее известные и широко применяемые – методы Зива-Лемпела (подразделы 4.3 и 4.4). Их описанию предшествует описание так называемых «монотонных» кодов. Эти коды входят как составная часть в один из алгоритмов Зива-Лемпела. В подразделе 4.3. описаны метод интервального кодирования и метод «стопка книг». Эти способы кодирования тоже полезны для понимания алгоритмов Зива-Лемпела. Кроме того, они имеют самостоятельное значение, в частности, они часто используются при кодировании видеоинформации. Метод «стопка книг» применяется также в рамках алгоритма Барроуза-Уилера. Далее мы приводим описание алгоритмов-рекордсменов по сжатию – алгоритма предсказания по частичному совпадению и алгоритма Барроуза-Уилера. В завершение раздела мы обсудим критерии сравнения эффективности архиваторов и приведем характеристики лучших из них.

Точный математический анализ практических алгоритмов кодирования сложен и выходит за рамки данного курса. Однако, как мы увидим из описания алгоритмов, они являются естественным развитием асимптотически эффективных теоретико-информационных методов, описанных в предыдущих разделах.

4.1. Монотонные коды

Все рассмотренные в разделе 3 универсальные кодеры, однопроходные и двухпроходные, используют при кодировании полученные тем или иным способом оценки вероятностей букв источника. Если объем алфавита очень велик (например, если в качестве алфавиты рассматриваются не отдельные

буквы, а пары, тройки букв), то хранение и модификация информации о каждой букве потребует значительных затрат памяти и времени. В этих случаях достаточно эффективными оказываются очень простые методы универсального кодирования, которым посвящен данный параграф.

Последовательность букв источника будет преобразовываться в последовательность натуральных чисел, поэтому нам понадобятся префиксные коды бесконечного (счетного) объема. Префиксный код множества натуральных чисел $\mathrm{N}=\{1,2,\ldots\}$ мы будем называть *монотонным кодом,* если для любых $i,j\in\mathrm{N}$, $i<j$, длины соответствующих кодовых слов l_i и l_j удовлетворяют неравенству $l_i\le l_j$.

Приведем несколько примеров монотонных кодов.

Пример 4.1.1. Унарный код.

Напомним, что запись вида 0^m или 1^m означает соответственно серию из m нулей или единиц. Унарный код сопоставляет числу i двоичную комбинацию вида $1^{i-1}0$. Например, унарными кодами чисел 1, 2 и 3 являются последовательности unar(1)=0, unar(2)=10 и unar(3)=110 соответственно. Длина кодового слова для числа i равна $l_i=i$. ■

Нетрудно проверить, что унарный код оптимален, если числа i распределены по геометрическому закону

$$p_i=(1-\alpha)^{i-1}\alpha,\quad i=1,2,\ldots$$

с параметром $\alpha=1/2$.

Для значений $\alpha<1/2$ более эффективен код Голомба.

Пример 4.1.2. Код Голомба [Gol66]**.**

Введем параметр $T=2^m$. Код Голомба для числа i состоит из двух частей. Первая часть – унарный код числа $\lfloor i/T\rfloor$, вторая часть – двоичная запись в виде последовательности длины m остатка от деления i на T. Очевидно, длина кода Голомба для числа i равна $l_i=\lfloor i/T\rfloor+m$. Например, при $m=3$ кодом Голомба числа 21 будет последовательность 001101. ■

В технической литературе код Голомба иногда называют кодом Райса.

Пример 4.1.3. Код Галлагера-ВанВухриса [GV75]**.**

Естественное обобщение кода Голомба на случай, когда параметр T не является степенью двойки – код Галлагера-ВанВухриса. Если $\log T$ – не целое число, то для представления некоторых (меньших) значений остатка от деления i на T используется $m=\lfloor\log T\rfloor$ бит, для остальных (больших) используется последовательность из $m+1$ бит. Например, при $T=5$ кодовыми словами для значений остатков 0, 1, 2, 3 и 4 будут последовательности 00, 01, 10, 110 и 111 соответственно.

Код Галлагера-ВанВухриса оптимален, если параметр T связан с параметром геометрического распределения α соотношением

$$(1-\alpha)^T+(1-\alpha)^{T+1}\le 1<(1-\alpha)^{T-1}+(1-\alpha)^T.\blacksquare \qquad (4.1.1)$$

Благодаря чрезвычайно простой схеме кодирования и достаточно высокой эффективности коды Голомба и Галлагера-ВанВухриса часто применяют при кодировании аудио и видеосигналов. Для применения в схемах универсального кодирования они не очень хороши. Во-первых, нужно знать (или оценивать) значение параметра T, во-вторых, длина кодового слова l_i, как и для унарного кода, линейно растет с увеличением числа i и для некоторых распределений

кодирование может быть неэффективным.

Пример 4.1.4. Код Левенштейна [Lev66]**.**

Этот код проще всего объяснить на конкретном примере. Предположим, что нужно передать число $i = 21$. Двоичное представление этого числа имеет вид 10101. Непосредственно использовать при кодировании двоичные представления натуральных чисел нельзя, такой код не будет префиксным. Самый простой выход состоит в том, чтобы приписать в начале слова префикс, указывающий длину двоичной записи числа (в данном случае это число 5). Если это число закодировать префиксным, например, унарным, кодом и приписать слева к двоичной записи числа, то код получится однозначно декодируемым. В данном примере для числа 21 получим кодовое слово 1111010101. В общем случае длина двоичного представления будет равна $2\lceil \log i \rceil$.

Будем шаг за шагом улучшать этот способ кодирования. Заметим, что первая значащая цифра двоичной записи числа – всегда 1. Ее можно не передавать, декодер сам допишет недостающую единицу, если будет знать длину двоичной записи числа. Обозначим через $\mathrm{bin}'(i)$ двоичную запись натурального числа i без первой единицы, прямыми скобками обозначается длина двоичной последовательности. Итак, простой монотонный код числа i имеет следующую структуру

$$\mathrm{mon}(i) = (\mathrm{unar}(|\mathrm{bin}'(i)| + 1), \mathrm{bin}'(i)). \qquad (4.1.2)$$

Длины кодовых слов равны

$$l_i = 2\lfloor \log i \rfloor + 1. \qquad (4.1.3)$$

Чтобы сделать запись еще короче, с длиной двоичной записи можно поступить так же, как и с самим числом, т.е. передать его значащие разряды (кроме первой единицы), затем длину двоичной записи числа значащих разрядов и т.д. Итерации продолжаются, пока не останется 1 значащий разряд. Чтобы декодирование было однозначным, достаточно приписать префикс, содержащий закодированное префиксным кодом число итераций. Заметим, что минимальное число итераций равно 0 (при кодировании числа 1). Поэтому в качестве префиксного кода можно выбрать унарный код увеличенного на 1 числа операций. Полученное кодовое слово будет кодовым словом *кода Левенштейна.*

Например, для числа 21 вычисляется $\mathrm{bin}'(21) = 0101$, затем $\mathrm{bin}'(4) = 00$, $\mathrm{bin}'(2) = 0$. Число итераций равно 3, поэтому кодовое слово кода Левенштейна имеет вид lev(21)=(1110)(0)(00)(0101)= 11100000101.

Декодер кода Левенштейна, декодируя унарный код, узнает, что итераций было 3. Прочитав один значащий разряд (0) и дописав к нему в начало 1, получает последовательность 10. Это означает, что на предпоследней итерации длина числа была 2. Прочитав 2 разряда и дописав слева 1, получает 100. Теперь декодер считывает 4 разряда и дописывает слева 1. Получается последовательность 10101, ей соответствует число 21. Поскольку это уже последняя 3-я итерация, число 21 является результатом декодирования. ■

В таблице 4.1.1. приведены кодовые слова, полученные по правилу (4.1.2) и кодовые слова кода Левенштейна для некоторых чисел натурального ряда и длины кодовых слов.

Пример 4.1.5. Упрощенный код Левенштейна (Код Элайеса).

Мы опишем более простой код, приведенный в работе Элайеса [El87]. Числу $i=1$ припишем кодовое слово elias(1)=0. Для чисел $i>1$ кодовые слова вычисляются по следующем правилу:

elias(i)=(unar(|bin'(|bin'(i)|)|+2) , bin'(|bin'(i)|) , bin'(i)) (4.1.4)

То есть, кодовое слово состоит из 3 частей. Справа (в третьей части) записываем двоичное представление числа без первой единицы. Ей предшествует вторая часть, в которой пишется двоичное представление длины третьей части, тоже без первой единицы. В первой части записан унарный код увеличенной на 2 длины второй части кодового слова.

Например, для числа 21 кодовое слово имеет вид elias(21) = (1110)(00)(0101)=1110000101. ■

Длина кодового слова кода Элайеса для произвольного числа i равна

$$l_i = \begin{cases} 1, & i=0; \\ \lfloor \log i \rfloor + 2\lfloor \log \lfloor \log i \rfloor \rfloor + 2, & i>1; \end{cases} \qquad (4.1.5)$$

и, следовательно, удовлетворяет неравенству

$$l_i \le \log i + 2\log(1+\log i) + 2\,,\ i=1,2,... \quad . \qquad (4.1.6)$$

Во втором слагаемом добавлена 1 под знаком логарифма для того, чтобы придать смысл этому выражению при $i=1$.

Таблица 4.1.1

Монотонные коды некоторых чисел натурального ряда

i	Монотонный код (4.1.2)		Код Левенштейна		Код Элайеса	
	Mon(i)	l_i	lev(i)	l_i	Elias(i)	l_i
1	0	1	0	1	0	1
2	100	3	100	3	100	3
3	101	3	101	3	101	3
4	11000	5	110000	6	110000	6
…	...	5	...	6	...	6
7	11011	5	110011	6	110011	6
8	1110000	7	1101000	7	1101000	7
…	...	7	...	7	...	7
15	1110111	7	1101111	7	1101111	7
16	111100000	9	11100000000	11	1110000000	10
...	...	9	...	11	...	10
31	111101111	9	11100001111	11	1110001111	10
32	11111000000	11	111000100000	12	11100100000	11
...	...	...	...	12	...	11
6	111110111	11	111000111111	1	111001111	1

3	11			2	11	1
2^{15}	$1^{15}00^{15}$	33	$1^{4}01111110^{15}$	26	$1^{5}011110^{15}$	25
2^{102}	$1^{1023}00^{1023}$	2047	$1^{4}010011^{9}0^{1023}$	1041	$1^{10}01^{9}10^{1023}$	1043

Для того чтобы численно сравнить три монотонных кода, обратимся к Таблице 4.1.1. Как ни странно, простейший код (4.1.2) в большинстве случаев дает наилучший результат, но для этого кода длина кодовых слов растет с увеличением i быстрее, чем для двух других кодов. Это видно, в частности, из сравнения формул (4.1.3) и (4.1.5). Коды Левенштейна и Элайеса практически эквивалентны, выигрыш кода Левенштейна проявляется только при астрономически больших значениях i.

Следующая теорема проясняет роль монотонных кодов в универсальном кодировании источников.

Теорема 4.1.1. Пусть случайная величина i принимает значения из множества чисел натурального ряда и распределение вероятностей случайной величины удовлетворяет условию: $p_i \le p_j$ если $i > j$. Тогда при использовании кода Элайеса средняя длина кодовых слов $\bar{l}$ удовлетворяет неравенству

$$\bar{l} \le H(1 + o(H)),$$

где через H обозначена энтропия случайной величины i, и $o(H) \to 0$ при $H \to \infty$.

Доказательство. Прежде всего, из упорядоченности вероятностей по убыванию и условия нормировки вероятностей вытекают неравенства

$$p_i \le \frac{1}{i}, \; i \le \frac{1}{p_i}, \; i = 1,2,... \; . \tag{4.1.7}$$

Из (4.1.6) имеем

$$\bar{l} = \sum_{i=1}^{\infty} l_i p_i \le \sum_{i=1}^{\infty} p_i (\log i + 2\log(1 + \log i) + 2) \le$$

$$\le \sum_{i=1}^{\infty} p_i \log \frac{1}{p_i} + 2\sum_{i=1}^{\infty} p_i \log\left(1 + \log \frac{1}{p_i}\right) + 2 \le H + 2\log(1 + H) + 2. \tag{4.1.8}$$

Здесь первое неравенство основано на (4.1.6), второе использует (4.1.7). Последнее неравенство основано на выпуклости вверх функции $\log x$. Из (4.1.8) следует утверждение теоремы. ■

Теорема 4.1.1. утверждает, что монотонный код с длиной i-го кодового слова порядка $\log i + 2\log\log i$ обеспечивает достаточно эффективное кодирование для любого источника с монотонно убывающими вероятностями букв. Поскольку буквы всегда можно перенумеровать, это означает, что при кодировании можно не знать вероятности букв, достаточно знать, как упорядочены вероятности букв.

Отметим, что за отсутствие точной информации о вероятностях букв мы платим избыточностью кодирования. При точно известных вероятностях побуквенное кодирование имеет избыточность не больше 1. Избыточность универсального монотонного кода имеет порядок $2\log H + 2$, что может быть приемлемым только при очень больших значениях энтропии источника H.

Отметим также простоту реализации кодирования. В отличие, например, от

кода Хаффмена, нет необходимости в хранении таблиц кодовых слов, кодирование и декодирование сводятся к вычислениям по приведенным выше достаточно простым формулам.

4.2. Интервальное кодирование и метод «стопка книг»

В этом параграфе мы рассмотрим два способа взаимно однозначного преобразования последовательности букв источника в последовательность чисел натурального ряда. При определенных условиях (например, при сильной зависимости букв) побуквенное кодирование преобразованной последовательности оказывается заметно более эффективным, чем кодирование исходной последовательности источника.

Начнем с *интервального кодирования.*

Пусть $X = \{0,1,...,M-1\}$ – алфавит источника и на выходе источника наблюдается последовательность $x_1, x_2, \ldots$. Алгоритм описывается рекуррентно. Предположим, что начальная часть последовательности $\mathbf{x}_1^{n-1} = (x_1,...,x_{n-1})$ уже закодирована и передана и, следовательно, известна декодеру. Вместо буквы x_n передается длина интервала (количество букв) между предыдущим и текущим появлением данной буквы. Иными словами, кодер вычисляет и передает минимальное число $r_n = r$ такое, что $x_{n-r} = x_n$. Тем самым исходная последовательность $x_1, x_2, \ldots$ преобразуется в последовательность чисел $r_1, r_2, \ldots$

Для завершения описания алгоритма нужно определить правило вычисления интервалов для тех букв, которые не встречались в последовательности $\mathbf{x}_1^{n-1}$. Проще всего условиться о том, что первой передаваемой букве предшествовали все буквы алфавита в заранее согласованном (для определенности, в алфавитном) порядке.

Опишем *метод «стопка книг».*

Этот метод имеет несколько названий и переоткрывался разными авторами. По-видимому, первым его описал и проанализировал его эффективность Б. Я. Рябко [Ry80]. Заметно позже в точности тот же алгоритм описан в [BSTW84] под названием «move-to-front coding». Именно под таким названием он чаще всего упоминается в технической литературе. Еще раз он открыт в [El87] и назван «recency rank coding». Это последнее название, которое можно перевести как «кодирование степени новизны», наиболее точно отражает суть дела.

Как и при кодировании длин интервалов, предположим, что начальная часть последовательности $\mathbf{x}_1^{n-1} = (x_1,...,x_{n-1})$ уже закодирована и передана и, следовательно, известна декодеру. Но теперь вместо буквы x_n передается количество *различных* букв между предыдущим и текущим появлением данной буквы. Иными словами, кодер вычисляет и передает минимальное число $r_n = r$ такое, что $x_{n-r} = x_n$. Затем вычисляется число d_n различных букв в последовательности $\mathbf{x}_{n-r+1}^{n-1}$. Тем самым исходная последовательность $x_1, x_2, \ldots$ преобразуется в последовательность чисел $d_1, d_2, \ldots$

Для передачи тех букв, которые не встречались в $\mathbf{x}_1^{n-1}$, можно использовать стратегию, описанную выше для интервального кодирования.

Пример 4.2.1. Пусть источник с алфавитом $X = \{a,b,c\}$ породил последовательность *cabbbabbac* . Рассмотрим работу двух методов кодирования

в предположении, что по умолчанию кодеры (и декодеры) считают, что этой последовательности предшествовала последовательность abc. Тогда при интервальном кодировании будет получена последовательность 0,3,3,0,0,3,1,0,2,9. При кодировании с помощью стопки книг получим последовательность 0,2,2,0,0,1,1,0,1,2. ■

Поясним происхождение названия «стопка книг». Предположим, что студент, готовясь к экзамену, сложил книги стопкой на столе. Всякий раз, выбирая из стопки нужную книгу, после использования, он не возвращает книгу на свое место, а кладет ее сверху. Верхняя книга теперь имеет наименьший номер, а номера книг, которые раньше находились выше нее, увеличились на 1. Сколько книг находятся выше данной? Ровно столько, сколько различных книг использовалось между текущим и предыдущим использованием данной книги. Таким образом, последовательность $d_1, d_2, \ldots$ это как раз и есть последовательность номеров книг, извлекаемых из стопки.

Пример 4.2.2. Применим «стопку книг» к пословице

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

Для этого вместо букв нужно подставить их номера в таблице ASCII-кода. В частности, пробелу соответствует число 32, а буквам A … Z – числа 65 … 90.

Результатом будет последовательность чисел

74, 72, 35, 88, 73, 3, 72, 71, 80, 1, 81, 86, 6, 76, 4, 3, 6, 86, 3, 10, 10, 3, 3, 6, 87, 83, 9, 6, 6, 7, 3, 8, 81, 9, 9, 9, 9, 7, 2, 5, 3, 10, 8, 3, 10, 10, 3, 13, 6, 13

Применим универсальный монотонный код, описываемый формулой (4.1.2). Затраты на передачу последовательности составят 332 бита. С одной стороны, мы получили заметное сжатие по сравнению с исходными затратами, составляющими 50×8=400 бит. С другой стороны, сравнивая этот результат с другими, приведенными в таблице 3.9.1, видим, что данный способ кодирования несколько проигрывает тем, которые рассматривались в разделе 3. Этот проигрыш – адекватная плата за простоту реализации. ■

Нетрудно понять, почему описанные методы преобразования потоков данных могут быть полезны при универсальном кодировании. Чем больше вероятность буквы, тем короче интервалы между ее появлениями. После преобразования последовательности источника в последовательность интервалов или последовательность номеров в стопке книг наиболее вероятными будут малые числа и менее вероятными – большие. Тем самым создаются предпосылки для успешного применения монотонных кодов, описанных в предыдущем параграфе. Из описания двух алгоритмов следует, что с точки зрения применения монотонных кодов метод стопки книг предпочтительнее, поскольку порождает числа, меньшие или равные числам, порождаемым интервальным кодером.

Ниже мы приводим верхнюю оценку средней скорости интервального кодирования, поскольку его анализ существенно проще. Понятно, что эта же граница верна и для сжатия стопкой книг.

Теорема 4.2.1. Пусть $H(X)$ – энтропия одномерного распределения дискретного стационарного источника. Средняя скорость интервального кодирования в сочетании с использованием кода Элайеса удовлетворяет неравенству

$$\overline{R} \le H(X)(1 + o(H(X))),$$

где $o(H) \to 0$ при $H \to \infty$.

Доказательство. Обозначим через $r_i(a)$ длину интервала между i-м и

$(i-1)$-м появлениями буквы $x=a$ и через $l_i(a)$ соответствующие затраты на передачу буквы. В соответствии с оценкой (4.1.6)

$$l_i(a) \le \log r_i(a) + 2\log(1+\log r_i(a)) + 2. \qquad (4.2.1)$$

Обозначим через $\bar{l}(a)$ и $\bar{r}(a)$ соответственно средние (по множеству последовательностей источника) затраты на передачу буквы a и среднюю длину интервала между появлениями буквы a. В силу выпуклости логарифма из (4.2.1) после усреднения получаем

$$\bar{l}(a) \le \log \bar{r}(a) + 2\log(1+\log \bar{r}(a)) + 2. \qquad (4.2.1)$$

В соответствии с леммой Каца (Кас) (ее доказательство приведено ниже) имеет место неравенство

$$\bar{r}(a) \le \frac{1}{p(a)}. \qquad (4.2.2)$$

Подставим эту оценку в (4.2.1) и усредним по всем буквам алфавита. Далее снова примем во внимание выпуклость логарифма. В результате получим

$$\bar{R} = \sum_a p(a)\bar{l}(a) \le H(X) + 2\log\bigl(1+H(X)\bigr) + 2.$$

Отсюда следует утверждение теоремы. ■

Итак, преобразование потока данных с помощью интервального кодирования или кодирования по методу стопки книг с последующим применением кода Левенштейна или кода Элайеса гарантирует достаточно эффективное универсальное кодирование при очень простой реализации кодера и декодера. Эти методы не требуют накопления статистических данных об источнике и построения соответствующего кода источника. Подобные идеи лежат в основе алгоритмов Зива-Лемпела, изучаемых в следующих параграфах.

Докажем лемму Каца. Нам понадобится вспомогательное тождество, которое мы сформулируем в виде отдельной леммы.

Лемма 4.2.1. Пусть случайная величина i принимает значения из множества натуральных чисел $\{1,2,...\}$ в соответствии с распределением вероятностей $\{p_i, i=1,2,...\}$. Тогда

$$\mathrm{M}[i] = \sum_{j=1}^{\infty} \mathrm{P}(i \ge j). \qquad (4.2.3)$$

Доказательство. Напомним, что $\mathrm{P}(i \ge j) = \sum_{i=j}^{\infty} p_i$. Нетрудно заметить, что сумма в правой части содержит слагаемое p_i ровно i раз. Следовательно, эта сумма равна математическому ожиданию i. ■

В теории вероятностей для дискретного случайного процесса $x_0, x_1, ...$ длины интервалов между одинаковыми значениями процесса $x_t = a$ называют временем возвращение процесса в состояние a. Более точно, время возвращения в состояние a для стационарного процесса определяется как

$$r = \min\{k \ge 1 : x_0 = x_k = a\}. \qquad (4.2.4)$$

Распределение вероятностей времени возвращения в a определяется формулой

$$q_a(r) = \mathrm{P}(x_1, ..., x_{k-1} \ne a, x_k = a \mid x_0 = a), \qquad (4.2.5)$$

соответственно среднее время возвращения в a равно

$$\bar{r}_a = \sum_{r=1}^{\infty} r q_a(r). \qquad (4.2.6)$$

Лемма 4.2.2. (Лемма Каца). Для дискретного стационарного источника такого, что $p(a) > 0$ справедливо соотношение

$$\bar{r}_a p(a) = \mathrm{P}(x_n = a \text{ хотя бы для одного } n,\ n = 0,1,...). \qquad (4.2.7)$$

В частности, для эргодического источника

$$\bar{r}(a) = \frac{1}{p(a)}. \qquad (4.2.8)$$

Доказательство. Рассмотрим произведение в левой части (4.2.7). Воспользовавшись леммой 4.2.1, получим

$$r_a p(a) = p(a)\sum_{t=1}^{\infty} r q_a(r) = p(a)\sum_{t=1}^{\infty} Q_a(t), \qquad (4.2.9)$$

где использовано обозначение

$$Q_a(t) = \sum_{j=t}^{\infty} q_a(j) = \mathrm{P}(r \ge t).$$

Событие $r \ge t$ (время возвращения в a не меньше t) имеет место в том случае, когда вслед за буквой a порождены t букв, ни одна из которых не совпадает с a, т.е.

$$Q_a(t) = \mathrm{P}(x_1,...,x_t \ne a \mid x_0 = a).$$

Подстановка этого выражения в (4.2.9) дает

$$r_a p(a) = \mathrm{P}(x_0 = a)\sum_{t=1}^{\infty} \mathrm{P}(x_1,...x_t \ne a \mid x_0 = a) = \sum_{t=1}^{\infty} \mathrm{P}(x_0 = a, x_1,...x_t \ne a).$$

Первое равенство основано на стационарности источника, второе – на определении условной вероятности.

Введем в рассмотрение еще одно распределение вероятностей на множестве последовательностей источника. Это новое распределение соответствует инверсии во времени последовательностей исходного источника, обозначим его как $\tilde{p}(\mathbf{x}), \mathbf{x} \in X^n, n = 1,2,...$. Для произвольного n и любой последовательности $\mathrm{x} = (x_1,...,x_n)$ обозначим через $\bar{\mathbf{x}}$ последовательность, полученную из $\mathbf{x}$ инвертированием порядка следования символов, т.е. $\bar{\mathbf{x}} = (x_n,...,x_1)$. По определению $\tilde{p}(\bar{\mathbf{x}}) = p(\mathbf{x})$. Вероятности, вычисленные относительно этого распределения будем обозначать как $\tilde{\mathrm{P}}(\cdot)$. Из последнего равенства с учетом стационарности инверсного процесса имеем

$$r_a p_a(t) = \sum_{t=1}^{\infty} \tilde{\mathrm{P}}(x_0,...,x_{t-1} \ne a, x_t = a) =$$

$$= \tilde{\mathrm{P}}(x_t = a \text{ хотя бы для одного } t,\ t = 1,2,...).$$

Вероятности появления хотя бы одной буквы a для исходного процесса и инверсного одинаковы, тем самым доказано первое утверждение леммы. Для любого эргодического процесса и любой буквы a такой, что $p(a) > 0$ эта вероятность равна 1, из чего следует второе утверждение. ■

4.3. Метод скользящего словаря (ZL-77)

Этот алгоритм после его опубликования Зивом и Лемпелом в статье [ZL77] был многократно модифицирован, некоторые модификации получили самостоятельные названия с аббревиатурами вида ZLX, где X – первая буква

имени автора модификации. В конце параграфа мы коротко обсудим практические аспекты применения алгоритма, в частности, простые усовершенствования, повышающие коэффициент сжатия. Начнем с формального описания алгоритма.

Параметром алгоритма является длина «окна наблюдения» W. Эту величину можно также интерпретировать как «объем скользящего словаря».

Пусть $X = \{0,1,...,M-1\}$ – алфавит источника и на выходе источника наблюдается последовательность $x_1, x_2, \ldots$. Алгоритм описывается рекуррентно. Предположим, что начальная часть последовательности $\mathbf{x}_1^n = (x_1,...,x_n)$ уже закодирована и передана и, следовательно, известна декодеру. Выполняются следующие вычисления.

Алгоритм ZL-77.

1. Находим максимальное l такое, что для некоторого d, $d \in \{1,...,W\}$ последовательность $\mathbf{x}_{n+1}^{n+l}$ (т.е. последовательность из l следующих символов) совпадает с некоторой последовательностью $\mathbf{x}_{n-d+1}^{n-d+l}$ длины l, наблюдавшейся в $\mathbf{x}_{n-W+1}^{n}$.

2. В случае успеха ($l > 0$),

к кодовой последовательности дописывается

- символ 1, указывающий на то, что найдена нетривиальная последовательность $\mathbf{x}_{n+1}^{n+l} = \mathbf{x}_{n-d+1}^{n-d+l}$, $d \in \{1,...,W\}$;
- последовательность из $\lceil \log W \rceil$ бит, представляющая собой двоичную запись числа d;
- число l, записанное в виде кодового слова некоторого неравномерного префиксного кода.

При этом значение n увеличивается на l и кодер переходит к кодированию следующих символов источника (шаг 1).

В противном случае ($l = 0$) передается

- символ 0, указывающий на то, что буква x_{n+1} не встречалась в окне $\mathbf{x}_{n-W+1}^{n}$,
- последовательность из $\lceil \log M \rceil$ бит, представляющая собой двоичную запись номера буквы x_{n+1} в алфавите X.

При этом значение n увеличивается на 1 и кодер переходит к кодированию следующих символов источника (шаг 1). ■

Заметим, что в начале работы алгоритма, т.е. при $n < W$ фактическая длина окна равна n и для передачи числа d достаточно $\lceil \log n \rceil$ бит (вместо $\lceil \log W \rceil$ бит).

Итак, кодер алгоритма ZL-77 хранит в памяти скользящий словарь объема W. Словами словаря служат всевозможные подпоследовательности следующих друг за другом букв, содержащиеся в последних W буквах источника. При поступлении на вход кодера новых букв кодер находит как можно более длинную последовательность, уже имеющуюся в словаре. В канал передается флаг (1 или 0), указывающий на то, найдено или нет подходящее словарное слово. В случае успеха (флаг равен 1) словарное слово передается указанием удаления начала слова от текущей позиции и длины словарного слова. Расстояние до слова d передается равномерным кодом, длина слова –

некоторым неравномерным кодом. Например, можно воспользоваться любым из монотонных кодов раздела 4.2. Если же словарного слова не нашлось, передается значение флага равное 0 и за ним следует очередная буква источника, передаваемая «без кодирования».

Поясним работу этого несложного алгоритма примером.

Пример 4.3.1. Применим алгоритм ZL-77 к пословице

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

Условимся о том, что длины слов l кодируются монотонным кодом (4.1.2). Результаты работы алгоритма шаг за шагом отображены в таблице 4.3.1. В этой таблице bin(·) обозначает стандартный 8-битный код буквы, в графе «расстояние» в скобках указана текущая длина окна. От нее зависит число бит, отводимое на передачу расстояния между текущей позицией и найденным в таблице образцом. Суммарные затраты на передачу последовательности составили 257 бит. ■

Таблица 4.3.1

Метод скользящего словаря (ZL-77)

Шаг	Флаг	Последователь-ность символов	Расстояние d	Длина l	Кодовая после-довательность	Затраты (бит)
0	0	I	–	0	0 bin(I)=0	9
1	0	F	–	0	0 bin(F)	9
2	0	_	–	0	0 bin(_)	9
3	0	W	–	0	0 bin(W)	9
4	0	E	–	0	0 bin(E)	9
5	1	_	2(5)	1	1 010 0	5
6	0	C	–	0	0 bin(C)	9
7	0	A	–	0	0 bin(A)	9
8	0	N	–	0	0 bin(N)	9
9	1	N	0(9)	1	1 0000 0	6
10	0	O	–	0	0 bin(O)	9
11	0	T	–	0	0 bin(T)	9
12	1	_	6(12)	1	1 0110 0	6
13	0	D	–	0	0 bin(D)	9
14	1	O	3(14)	1	1 0011 0	6
15	1	_	2((15)	1	1 0010 0	6
16	1	A	8(16)	1	1 00100 0	7
17	0	S	–	0	0 bin(S)	9

18	1	_WE_	15(18)	4	1 01111 11000	11
19	1	W	2(22)	1	1 00010 0	7
20	1	O	8(23)	1	1 00100 0	7
21	0	U	–	0	0 bin(U)	9
22	0	L	–	0	0 bin(L)	9
23	1	D	12(26)	1	1 01100 0	7
24	1	_WE_	8(27)	4	1 00100 11000	11
25	1	S	13(28)	1	1 01101 0	7
26	0	H	–	0	0 bin(H)	9
27	1	OULD_	9(33)	5	1 001001 11001	12
28	1	DO_AS_WE_	24(38)	9	1 011000 1110001	14
29	1	CAN	40(47)	3	1 101000 101	10
Итого						257

Пример показывает, что алгоритм ZL-77 существенно выигрывает по сжатию по сравнению с логически более сложными методами раздела 3, Больше того, из таблицы 4.3.1 ясно видно, что эффективность алгоритма быстро растет по мере увеличения объема скользящего словаря.

Математический анализ алгоритма сложен и выходит за рамки данного пособия. Однако следующие рассуждения позволяют понять характер поведения алгоритма с увеличением объема словаря. Действительно, длина кодовой последовательности, сопоставляемой отрезку $\mathbf{x}_{n+1}^{n+l}$ составляет

$$L(\mathbf{x}_{n+1}^{n+l}) \approx 1 + \log(W) + 2\log l$$

бит. Эти биты потрачены на передачу l букв источника. Заметим, что с увеличением объема словаря увеличиваются длины последовательностей, которые кодер находит в словаре. Таким образом, при больших W (и, соответственно, l) «мгновенная скорость» кодирования

$$R(\mathbf{x}_{n+1}^{n+l}) \approx \frac{1+\log W + 2\log l}{l} = \frac{\log W}{l} + o(l),$$

где $o(l) \to 0$ при увеличении W и l. Теперь нужно установить связь между W и l при больших W. Понятно, что сочетание букв, имеющее вероятность p, встретится в словаре объема W примерно pW раз. (Это, в частности, видно из леммы Каца, которая устанавливает, что средние интервалы между такими событиями имеют длину $1/p$). Если $pW >> 1$, то, вероятнее всего, сочетание букв имеет продолжение, которое также имеется в словаре. Отсюда следует,

что типичные согласованные последовательности наблюдаются в окне длины W конечное число раз и что при больших W имеет место приближенное соотношение

$$p(\mathbf{x}_{n+1}^{n+l}) \approx \frac{K}{W},$$

где K - константа. Подставив это соотношение в выражение для скорости кодирования, получаем

$$R(\mathbf{x}_{n+1}^{n+l}) \approx \frac{1}{l}\log\frac{1}{p(\mathbf{x}_{n+1}^{n+l})} + \frac{\log(K)}{l} + o(l),$$

Усредняя по множеству последовательностей источника, приходим к выводу, что

$$\overline{R} = \mathrm{M}\left[R(\mathbf{x}_{n+1}^{n+l})\right] \underset{l\to\infty}{\to} H_\infty(X).$$

Можно оценить скорость убывания избыточности с увеличением W. Основной вклад в избыточность вносит слагаемое, связанное с необходимостью передачи длины согласованного отрезка, т. е. избыточность имеет порядок $(\log l)/l$. Поскольку асимптотически (для типичных последовательностей) W пропорционально $\log l$, то асимптотическое поведение избыточности кодирования с увеличением W определяется формулой

$$\overline{R} - H_\infty(X) \approx \frac{\log\log W}{\log W}.$$

Напомним, что скорость убывания избыточности универсального кодирования последовательности длины n для методов раздела 3 имеет порядок $\log n/n$. Это означает, что избыточность алгоритма ZL-77 убывает (асимптотически) намного медленнее. Тем не менее, соотношение между сложностью алгоритма и сжатием именно для этого алгоритма оказалось едва ли не самым лучшим среди известных на сегодняшний день алгоритмов.

Рассмотрим коротко некоторые аспекты практической реализации алгоритма и его основные модификации.

Обсудим сложность алгоритма. Прежде всего, очевидно, сложности кодирования и декодирования не равны. Вычислительная сложность декодирования очень низкая. Декодер непосредственно из битого потока получает информацию о том, начиная с какой буквы и какой длины последовательность букв он должен извлечь из памяти и выдать получателю.

Напротив, кодер для каждого из значений $l = 1,2,...$ должен повторить попытку поиска в окне длины W (типичные размеры окна составляют 2048 – 16384 символов) образцов, совпадающих с вновь поступившими буквами источника. Такая «прямолинейная» реализация алгоритма неприемлемо сложна. Применяемые в архиваторах кодеры для быстрого поиска образцов в словаре кодера используют хеширование.

Рассмотрим подробнее алгоритм работы быстрого кодера, использующего алгоритм ZL-77. Прежде всего, заметим, что с увеличением длины окна ссылки на последовательности из 1 или 2 символов становятся неэффективными. Такие последовательности проще передать по одной букве, затрачивая по 9 бит на букву, как это делается для вновь встретившихся букв. Трехбуквенные последовательности рассматриваются как двоичные последовательности длины 24, и для них вычисляется хеш-функция, отображающая последовательность

длины 24 в двоичную последовательность длины 12. Эта последовательность используется как адрес ячейки памяти, в которой хранится указание на место в окне, где последний раз встречалось такое же сочетание букв. Эта информация используется далее при кодировании, а на ее место записывается указание на текущую позицию.

Кодер, имея информацию о последнем появлении 3-буквенного сочетания, проверяет истинную длину совпадения. Далее кодер должен проверять другие места в окне, где были такие же сочетания букв. Некоторые быстрые кодеры этого не делают, теряя при этом в сжатии информации. Во многих задачах, особенно, при реализации кодеров на БИС, это вполне разумная плата за существенное упрощение и ускорение работы кодера. Если же требуется более высокое сжатие, необходимо для каждой позиции окна хранить и обновлять информацию о том, в какой точке окна в последний раз встретилось такое же сочетание букв, как и в текущей позиции.

С учетом этих упрощений алгоритм ZL-77 на сегодняшний день, пожалуй, самый быстрый и на основе его модификаций построены практически все архиваторы.

Рассмотрим теперь подходы к повышению эффективности кодирования. Напомним, что на каждом шаге кодирования формируются 3 случайных величины: двоичный флаг, ссылка, принимающая значения из конечного диапазона $\{1,\ldots, W\}$, и длина ссылки, которая теоретически может быть любым натуральным числом, но на практике большие значения длин встречаются редко.

Первый путь к улучшению кодирования – использование более эффективных кодов для этих величин, чем, рассмотренные выше коды. В частности, можно использовать адаптивное кодирование на основе арифметического кодирования или кодов Хаффмена. Эта возможность используется во всех архиваторах.

Поучительной в этом смысле является модификация ZL-77, предложенная Фиала и Гриине [FG89]. Ее называют алгоритмом ZLFG. Рассмотрим сначала самый простой вариант алгоритма ZLFG.

Как и в ZL-77, на каждом шаге кодирования формируются кодовые комбинации двух типов, один из них соответствует ссылке на уже содержащиеся в окне последовательности букв, другой тип – непосредственной передаче последовательности букв источника.

Алгоритм ZLFG.

1. Находим максимальное l из диапазона $\{3,\ldots 17\}$ такое, что для некоторого d, $d \in \{1,\ldots,W\}$ последовательность $\mathbf{x}_{n+1}^{n+l}$ (т.е. последовательность из l следующих символов) совпадает с некоторой последовательностью $\mathbf{x}_{n-d+1}^{n-d+l}$ длины l, наблюдавшейся в $\mathbf{x}_{n-W+1}^{n}$.

2. В случае успеха ($l > 2$),

к кодовой последовательности дописывается

- число $l-2$, представленное в виде ненулевой двоичной последовательности длины 4 (числам 3,…,17 соответствуют комбинации 0001,…, 1111);
- последовательность из $\lceil \log W \rceil$ бит, представляющая собой двоичную запись числа d;

При этом значение n увеличивается на l и кодер переходит к кодированию следующих символов источника (шаг 1).

В противном случае ($l \le 2$) передается

- последовательность 0000, указывающая на то, что буква x_{n+1} не встречалась в окне $\mathbf{x}_{n-W+1}^{n}$;
- последовательность из 4 бит, представляющая собой двоичную запись числа букв L, поступающих на выход кодера непосредственно с выхода источника (без кодирования), число L принимает значения из множества $\{1,...,16\}$, этим значениям соответствуют двоичные последовательности 0000,…,1111;
- последовательность из L букв источника, передаваемых без кодирования.

При этом значение n увеличивается на L и кодер переходит к кодированию следующих символов источника (шаг 1). ▪

В этом алгоритме длины частей кодовых слов равные 4 выбраны для того, чтобы сделать удобной реализацию алгоритма на компьютере с байтовой организацией памяти. Если выбрать $W = 2^{12} = 4096$, то в любой из двух веток алгоритма длина кодовой комбинации кратна 8, т.е. составляет целое число байт.

Идея алгоритма понятна – уменьшить затраты битов на передачу тех отдельных букв источника или пар букв, которые нецелесообразно передавать с помощью ссылок на информацию, содержащуюся в скользящем словаре. Насколько эффективно такое решение, показывает следующий пример.

Пример 4.3.1. Применим алгоритм ZLFG к пословице

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

Чтобы адекватно сравнивать с описанной выше версией алгоритма ZL-77, на начальном этапе работы алгоритма будем учитывать, что при $n < W$ фактическая длина окна равна n и для передачи числа d достаточно $\lceil \log n \rceil$ бит (вместо $\lceil \log W \rceil$ бит). Результаты работы алгоритма по шагам приведены в таблице 4.3.2.

Таблица 4.3.2

Пример использования алгоритма ZLFG

Шаг	Длина совпадения	Расстояние до образца	Число «новых» букв	Кодовые символы	Передан-ные буквы	Затраты (бит)
1	0	–	16	0000 1111bin(I…O)	IF_WE_CANNOT_DO	8+8×15=128
2	1(<3)	–	3	0000 0010bin(_AS)	_AS	8+8×3=32
3	4	15(18)	–	0010 01111	_WE_	4+5=9
4	1(<3)	–	5	0000 0001bin(WOULD)	WOULD	8+5×8=48

5	5	8(27)	–	0011 01000	_WE_	4+5 =9
6	–	–		0000 0001bin(SH)	SH	8+2×8=24
7	5	9(33)	–	0011 001001	OULD_	4+6 =10
8	9	24(38)	–	0111 011000	DO_AS_WE_	4+6=10
9	3	40(47)	–	0001 101000	CAN	4+6=10
Итого						280

Сопоставляя эту таблицу с таблицей 4.3.1, нетрудно убедиться в том, что на этом коротком примере алгоритм ZLFG проиграл алгоритму ZL-77, в основном, потому, что ссылки на короткие последовательности (длины 1…3) оказались достаточно эффективными при коротком окне. Для длинных последовательностей алгоритм ZLFG в среднем эффективнее ZL-77. ▪

Вторая возможность увеличения сжатия – устранение избыточности кодирования, заведомо присущей описанному данному алгоритму. Поскольку в скользящем словаре имеются одинаковые последовательности, одной и той же последовательности букв источника могут быть сопоставлены различные кодовые слова. Поэтому некоторые кодовые слова просто никогда не используются.

В работе [FG89] описан способ хранения словаря в виде древовидной структуры. Ребрам дерева соответствуют последовательности букв. Ссылки разрешены только на узлы дерева. Поскольку число узлов меньше общего числа букв, уменьшаются затраты на передачу адреса слова в словаре.

В различных статьях и патентах можно найти много других методов усовершенствования алгоритма ZL-77. Задайте вашей любимой поисковой системе сети INTERNET ключевое слово «ZL-77» и вы найдете много интересной информации об этом алгоритме.

4.4. Алгоритм ЗЛВ (ZL-78)

Так сложилось, что из двух алгоритмов Зива-Лемпела второй алгоритм, описанный в работе [ZL78], поначалу казался более практичным и первым вошел в практику сжатия. В частности, он использован в стандарте V40bis для передачи данных по стандартным телефонным каналам общего пользования. С учетом модификации, предложенной Велчем в работе [W84], ZL-78 действительно выглядит элегантнее первого алгоритма. Ниже мы приводим описание этой модификации, называемой алгоритмом ZLW.

В процессе работы алгоритма кодер и декодер синхронно формируют словарь. Этот словарь на каждом шаге пополняется одним новым словом, которое до этого в словаре отсутствовало, но является продолжением на одну букву одного из слов словаря.

Пусть $X = \{0,1,...,M-1\}$ – алфавит источника и на выходе источника наблюдается последовательность $x_1, x_2, \ldots$. Алгоритм описывается рекуррентно. Предположим, что начальная часть последовательности $\mathbf{x}_1^n = (x_1,...,x_n)$ уже закодирована и передана. Предположим также, что при этом (и кодером и декодером) построен одинаковый словарь из c слов. Словарь образован словами, вообще говоря, различной длины из букв алфавита X. Для простоты описания алгоритма будем считать, что каждая из букв алфавита является

словом длины 1 и входит в состав словаря. (Все буквы могут быть записаны в словарь до начала работы алгоритма). На очередном шаге алгоритма выполняются следующие вычисления.

Алгоритм ZLW.

Находим максимальное l такое, что для некоторого j, $j \in \{1,...,c-1\}$ последовательность $\mathbf{x}_{n+1}^{n+l}$ (т.е. последовательность из l следующих символов) совпадает с j-м словом словаря. К кодовой последовательности дописывается двоичная последовательность длины $\lceil \log(c-1) \rceil$, представляющая собой двоичную запись номера словарного слова j. В словарь дописывается новое слово длины $l+1$ вида $\mathbf{x}_{n+1}^{n+l+1}$. Число n увеличивается на l, объем словаря c увеличивается на 1. Кодер переходит к кодированию следующих символов источника. ■

Заметим, что вновь дописанное в словарь слово состоит из уже имеющегося в словаре слова $\mathbf{x}_{n+1}^{n+l}$ и следующей за ним буквы источника x_{n+l+1}. Слова $\mathbf{x}_{n+1}^{n+l+1} = (\mathbf{x}_{n+1}^{n+l}, x_{n+l+1})$ в словаре нет, поскольку иначе длина совпадения l была бы на единицу больше, что противоречит алгоритму. Декодер получит из канала только указание на словарное слово $\mathbf{x}_{n+1}^{n+l}$. Этого, конечно, недостаточно для построения нового $(c+1)$-го слова $\mathbf{x}_{n+1}^{n+l+1}$, но недостающая буква x_{n+l+1} будет первой буквой словарного слова, найденного на следующем шаге кодирования. Таким образом, декодер будет строить словарь с опозданием на один шаг. Именно поэтому в описании алгоритма мы разрешаем выбирать слова с номерами не более $c-1$, и на передачу номера отводим $\lceil \log(c-1) \rceil$ бит.

Еще одно замечание касается инициализации алгоритма. При кодировании коротких последовательностей нецелесообразно инициализировать алгоритм, заполняя словарь всеми буквами алфавита источника. Разумнее использовать принцип «*esc*-кода». В начале работы алгоритма словарь пуст и в нем, в ячейке с нулевым номером хранится *esc*-код. В дальнейшем, при получении от источника буквы, отсутствующей в словаре, мы передаем *esc*-код, которому соответствует нулевая последовательность длины $\lceil \log(c-1) \rceil$, а вслед за *esc*-кодом – стандартный код новой буквы. Работа кодера поясняется следующим примером.

Пример 4.4.1. Применим алгоритм ZLW к пословице

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

Результаты работы алгоритма по шагам приведены в таблице 4.4.1.

Таблица 4.4.1

Пример использования алгоритма ZLW

Шаг	Словарь	Номер слова	Кодовая последовательность	Число бит
0	*esc*	–	–	–
1	I	0	bin(I)	8=8
2	F	0	bin(F)	log(1)+8=8
3	_	0	0bin(_)	$\lceil \log(2) \rceil$+8=9
4	W	0	00 bin(W)	$\lceil \log(3) \rceil$+8=10

5	E	0	00 bin(E)	$\lceil \log(4) \rceil$+8=10
6	_C	3	011	3
7	C	0	000bin(C)	3+8=11
8	A	0	000bin(A)	3+8=11
9	N	0	000bin(N)	3+8=11
10	N	0	000bin(N)	4+8=12
11	O	0	0000bin(O)	4+8=12
12	T	0	0000bin(T)	4+8=12
13	_D	3	0011	4
14	D	0	0000bin(D)	4+8=12
15	O_	11	1011	4=4
16	_A	3	0011	4
17	AS	8	1000	4
18	S	0	00000bin(S)	5+8
19	_W	3	00011	5
20	WE	4	00100	5
21	E_	5	00101	5
22	_WO	19	10011	5
23	OU	3	00011	5
24	U	0	00000bin(U)	5+8=13
25	L	0	00000bin(L)	5+8=13
26	D_	14	01110	5
27	_WE	19	10011	5
28	E_S	21	10101	5
29	SH	18	10010	5
30	H	0	00000bin(H)	5+8=13
31	OUL	23	10111	5
32	LD	25	11001	5
33	D_D	26	11010	5
34	DO	14	001110	6
35	O_A	15	001111	6
36	AS_	17	010001	6
37	_WE_	27	011011	6
38	_CA	6	000110	6
39	AN	8	001000	6
40	N	9	001001	6
Итого				299

Затраты на кодирование в данном примере оказались выше, чем для других способов, но, внимательно посмотрев на таблицу, мы видим, что к моменту окончания кодирования словарь уже содержит много очень полезных сочетаний букв, которые пригодились бы при кодировании дальнейшего текста. Таким образом, можно ожидать, что с увеличением длины кодируемой последовательности эффективность кодирования будет быстро нарастать. ■

Анализ алгоритма ZL-78 заметно проще, чем алгоритма ZL-77. В частности, в книге [CT91] приведено доказательство сходимости средней скорости кодирования к энтропии на сообщение. В данном пособии мы ограничимся нестрогим эвристическим рассуждением, позволяющим охарактеризовать асимптотическое поведение алгоритма с ростом длины кодируемой последовательности.

Предположим, что для достаточно большого числа n последовательность

$\mathbf{x}_1^n = (x_1,...,x_n)$ уже закодирована, и в процессе кодирования построен словарь объема c. По мере заполнения словаря в него включались последовательности длины $l = 1,2,...$. Разумно предположить, что словарь попадали главным образом все типичные последовательности длины каждой длины l. Как мы знаем из части 1, количество типичных последовательностей длины l приблизительно равно $2^{lH_l(X)}$, где $H_l(X) = H(X^l)/l$ – энтропия на сообщение последовательности длины l. Поэтому объем словаря связан с типичной длиной l очередной последовательности, включаемой в словарь, приближенным соотношением

$$c \approx \sum_{j=1}^{l} 2^{lH_l} .$$

С ростом l энтропии H_l убывают, но, начиная с некоторого значения, это убывание замедляется и значение H_l приближается к пределу $H = H_\infty(X) = H(X \mid X^\infty)$. Поэтому слагаемые в выражении для c можно рассматривать как возрастающую геометрическую прогрессию со знаменателем 2^H. Последнее слагаемое будет наибольшим, поэтому имеет место приближенное равенство

$$c \approx l2^{lH} .$$

Средняя скорость кодирования на данном шаге

$$R \approx \frac{\log c}{l} \approx H + \frac{\log l}{l} .$$

Поскольку величина l имеет порядок $\log c$, скорость кодирования и объем словаря (асимптотически при $c \to \infty$) связаны соотношением

$$R \approx H + \frac{\log\log c}{\log c} .$$

Это приблизительно такая же зависимость, какая имела место и для алгоритма ZL-77. Кодирование в ZL-78 на первый взгляд кажется более эффективным, поскольку формируемая кодовая последовательность, во-первых, не разбита на части, и, во-вторых, словарь состоит из несовпадающих словарных слов. Напротив, важное преимущество ZL-77 состоит в том, что в ZL-77 словарь (за исключением начального этапа работы алгоритма) имеет постоянный объем. В ZL-78 объем словаря увеличивается с каждым шагом, и, в конце концов, его объем превысит допустимый предел.

С точки зрения практического применения алгоритма ZLW важно ответить на два вопроса: как организовать хранение в памяти кодера и декодера большого словаря, составленного из слов различной длины и как поступать при заполнении этим словарем всего допустимого объема памяти.

Ответ на первый вопрос содержится в работе Велча [W84]. Поскольку словарь имеет древовидную структуру, то и хранить его нужно в виде дерева, сохраняя для каждого узла дерева адрес соседней вершины и адрес старшего потомка вершины. При такой структуре словаря поиск в словаре также становится достаточно быстрым. Помимо этого, дополнительное ускорение поиска возможно с помощью хеширования.

Второй вопрос сложнее. Тривиальными решениями проблемы являются

- прекращение наращивания словаря после того, как исчерпана допустимая память;
- полный сброс памяти и возвращение алгоритма к начальному состоянию.

Оба варианта имеют свои достоинства и недостатки. Можно предложить различные альтернативные варианты алгоритма или найти их с помощью Интернета.

4.5. Предсказание по частичному совпадению

Сейчас мы рассмотрим наиболее "логичный" алгоритм кодирования при неизвестной статистике источника. На первый взгляд он является почти тривиальным развитием адаптивного арифметического кодирования, которое обсуждалось в разделе 3.7. Разница состоит в том, что в процессе кодирования вместо безусловных вероятностей букв оцениваются их условные вероятности, при известном "контексте", т.е. при известных предшествующих буквах. Клеари и Виттену [CW84] принадлежит идея использовать контексты различной длины, зависящей от предшествующей закодированной последовательности данных. Этот метод кодирования получил название PPM (prediction by partial matching). Практическая реализация этой простой идеи порождает много вопросов, которые мы кратко обсудим в данном параграфе.

С момента опубликования исходной версии алгоритма было предложено очень много его модификаций. Все же любой из PPM-алгоритмов вкладывается в следующую схему. Предположим, что последовательность $\mathbf{x}_1^n = (x_1,...,x_n)$ первых n букв источника уже передана и предстоит передать символ x_{n+1}. При кодировании очередной буквы выполняются следующие шаги.

Идея алгоритма PPM

1. Находим контекст $\mathbf{x}_{n-d+1}^n$ наибольшей длины d, не превышающей заданной величины D. Под контекстом понимается последовательность $\mathbf{x}_{n-d+1}^n$ непосредственно предшествующая кодируемому символу и такая, что последовательность $\mathbf{x}_{n-d+1}^{n+1} = (\mathbf{x}_{n-d+1}^n, x_{n+1})$ уже встречалась ранее в переданной последовательности $\mathbf{x}_1^n$.
2. Для всех возможных значений символа x_{n+1} вычисляются условные вероятности символа при известном контексте.
3. Значение символа x_{n+1} кодируется арифметическим кодом в соответствии с вычисленным на шаге 2 условным распределением вероятностей и с учетом длины d контекста. ▪

Варианты алгоритма PPM отличаются друг от друга выбором максимальной длины контекста D, и , главное, способом выполнения шага 2, т.е. вычисления условной вероятности символа. По сути дела, нужно совместно кодировать истинную длину контекста d и букву x_{n+1}.

Естественной оценкой условной вероятности буквы $x_{n+1} = a$ после контекста $\mathbf{x}_{n-d+1}^n = \mathbf{c}$ кажется оценка вида

$$\hat{p}(a \mid \mathbf{c}) = \frac{N(\mathbf{c}, a)}{N(\mathbf{c})}.$$

В числителе этой формулы записано число появлений буквы a после контекста $\mathbf{c}$ в предшествующей последовательности символов $\mathbf{x}_1^n$, в знаменателе – общее число появлений $\mathbf{c}$ в $\mathbf{x}_1^n$. Однако, арифметический кодер не сможет воспользоваться этой формулой для тех букв a, которые не встречались с заданным контекстом, т.е. при $N(\mathbf{c},a)=0$. Чтобы исправить ситуацию, следует воспользоваться тем же подходом, что и при побуквенном адаптивном кодировании – ввести *esc*-символы, указывающие на то, что при заданном контексте данная буква не встречалась.

Итак, стратегия PPM-кодера обычно строится следующим образом.

Алгоритм PPM

1. Находим последовательность $\mathbf{x}_{n-d+1}^n$ наибольшей длины $d \le D$, такую, что точно такая же последовательность уже встречалась ранее в переданной последовательности $\mathbf{x}_1^n$. Выбираем ее в качестве контекста $\mathbf{c}$.
2. Если $N(\mathbf{c},x_{n+1})>0$, то оценивается условное распределение $\hat{p}(a\,|\,\mathbf{c}=\mathbf{x}_{n-d+1}^n)$ (формулы для оценки будут даны ниже) и буква x_{n+1} кодируется арифметическим кодом в соответствии с вычисленным распределением. Если же $N(\mathbf{c},x_{n+1})=0$, передается *esc*-символ (оценку его вероятности мы приведем ниже) и длина контекста d уменьшается на единицу.

   Рекурсия продолжается до тех пор, пока символ не будет передан, либо длина контекста не окажется равной 0. Это соответствует независимому кодированию символа x_{n+1} в соответствии с вычисленным по $\mathbf{x}_1^n$ безусловным распределением вероятностей символов алфавита X. Если буква x_{n+1} не встречалась в $\mathbf{x}_1^n$, то снова передается *esc*-символ и передается в соответствии с равномерным распределением вероятностей на множестве неиспользованных букв алфавита источника. ■

Итак, кодер старается передать букву арифметическим кодером в соответствии с вероятностями, определяемыми контекстом наибольшей возможной длины. Если это не удается сделать (буква еще не встречалась вслед за таким контекстом), передается *esc*-символ, контекст укорачивается на одну букву. В конечном счете, буква будет передана с учетом контекста меньшей (возможно, нулевой) длины, либо как "новая буква", если она не встречалась в уже закодированной последовательности букв

Чтобы окончательно конкретизировать алгоритм, нужно выписать формулы для вероятностей букв и *esc*-символов. Вопрос об адаптивном оценивании вероятностей букв мы уже рассматривали в разделе 3.7 в связи с адаптивным арифметическим кодированием. Перепишем оценки (3.7.3) и (3.7.6) так, чтобы использовать их в рамках алгоритма PPM. Оценки условных вероятностей букв и *esc*-символов для алгоритма A имеют вид

$$\hat{p}_n(a\,|\,\mathbf{c}) = \frac{\tau_n(\mathbf{c},a)}{\tau_n(\mathbf{c})+1}\ ,\ \ \tau_n(\mathbf{c},a)>0\,, \tag{4.5.1}$$

$$\hat{p}_n(esc\,|\,\mathbf{c}) = \frac{1}{\tau_n(\mathbf{c})+1}, \tag{4.5.2}$$

где $\tau_n(\mathbf{c})$ и $\tau_n(\mathbf{c},a)$ обозначают число последовательностей соответственно **c** и $(\mathbf{c},a)$, содержащихся в последовательности длины n. Аналогично изменяются оценки для алгоритма D:

$$\hat{p}_n(a\,|\,\mathbf{c}) = \frac{\tau_n(\mathbf{c},a) - 1/2}{\tau_n(\mathbf{c})}\ ,\ \ \tau_n(\mathbf{c},a) > 0\,, \tag{4.5.3}$$

$$\hat{p}_n(esc\,|\,\mathbf{c}) = \frac{M_n(\mathbf{c})}{2\tau_n(\mathbf{c})}, \tag{4.5.4}$$

где $M_n(\mathbf{c})$ – число различных букв, появившихся в последовательности длины n вслед за контекстом **c**.

Формулы (4.5.1)-(4.5.4) почти завершают описание двух версий алгоритма PPM: PPMA, предложенного в работе [CW84], и его модификации PPMD, описанной в [H93]. Осталось еще одно небольшое, но важное усовершенствование, введенное еще в самой первой работе [CW84]. Это усовершенствование называется техникой "исключений" (exclusions). Поясним идею этого метода примером. Предположим, что при кодировании текста на русском языке на некотором шаге контекст имеет вид "кор", а следующая за ним буква – бука "т", причем сочетание букв "корт" ранее не встречалось.

В соответствии с алгоритмом, предается *esc*-символ и контекст укорачивается на единицу, т.е. теперь контекстом будет "ор". Метод исключений рекомендует принять во внимание то, что не все сочетания "ор" следует использовать для подсчета условной вероятности буквы "т", а только те, которые не начинались с буквы "к". Действительно, ведь и кодер и декодер (благодаря передаче *esc*-символа) уже знают, что сочетание букв "корт" ранее не было. После такой корректировки знаменатель в оценке вероятности буквы (4.5.1) PPMA-кодера (или (4.5.3) для PPMD-кодера) уменьшится, а числитель останется прежним. Оценки вероятностей букв увеличатся, а значит затраты на передачу уменьшатся.

Следующий пример поможет глубже понять работу алгоритма PPM.

Пример 4.5.1. Применим алгоритмы PPM с параметром D=5 к пословице IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

Результаты работы алгоритмов по шагам приведены в таблицах 4.5.1. и 4.5.2. Знаки # в графе "контекст" соответствуют контекстам нулевой длины. Штрихом помечены те вероятности, которые подсчитаны с учетом исключений. В графе $\tau_n(\mathbf{c})$ представлено число появлений данного контекста и, если он укорачивался в процессе кодирования, то, через запятую, – число появлений укороченных контекстов. В графе $M_n(\mathbf{c})$ также через запятую приведены значения числа различных букв, следовавших за контекстом и его укорочениями. Рассмотрим несколько характерных шагов кодирования на примере алгоритма PPMA.

Шаг 6. Пробел встретился второй раз. Поскольку предшествующая ему буква E встречалась только один раз, контекст имеет нулевую длину. Для такого контекста пробел имеет вероятность 1/(5+1)=1/6.

Таблица 4.5.1

Пример использования алгоритма PPMA

Шаг	Буква	Контекст	$\tau_n(\mathbf{c})$	PPMA	
				$\hat{p}(esc \mid \mathbf{c})$	$\hat{p}(a \mid \mathbf{c})$
1	I	#	0	1	1/256
2	F	#	1	1/2	1/255
3	_	#	2	1/3	1/254
4	W	#	3	1/4	1/253
5	E	#	4	1/5	1/252
6	_	#	5		1/6
7	C	_	1,6	1/2×1/6'	1/251
8	A	#	7	1/8	1/250
9	N	#	8	1/9	1/249
10	N	#	9		1/10
11	O	N	1,10	1/2×1/9'	1/248
12	T	#	11	1/12	1/247
13	_	#	12		2/13
14	D	_	2,13	1/3×1/12'	1/246
15	O	#	14		1/15
16	_	O	1,15	1/2	3/15'
17	A	_	3,16	1/4	1/14'
18	S	A	1,17	1/2×1/17'	1/245
19	_	#	18		4/19
20	W	_	4		1/5
21	E	_W	1		1/2
22	_	_WE	1		1/2
23	W	_WE_	1,1,1,5	1/2×1'×1'	2/5'
24	O	_W	2,2,23	1/3×1'	2/22'
25	U	O	2,24	1/3×1/23'	1/244
26	L	#	25	1/26	1/243
27	D	#	26		1/27
28	_	D	1,27	1/2	6/27'
29	W	_	6		3/7
30	E	_W	3		2/4
31	_	_WE	2		2/3
32	S	_WE_	2,2,2,7,31	1/3×1'×1'×1/3'	1/25'
33	H	S	1,32	1/2×1/26'	1/242
34	O	#	33		3/34
35	U	O	3		1/4
36	L	OU	1		1/2
37	D	OUL	1		1/2
38	_	OULD	1		1/2
39	D	OULD_	1,1,1,1,8	1/2×1'×1'×1'	1/4'
40	O	_D	1		1/2
41	_	_DO	1		1/2
42	A	_DO_	1		1/2
43	S	_DO_A	1		1/2
44	_	DO_AS	1		1/2
45	W	O_AS_	1		1/2
46	E	_AS_W	1		1/2
47	_	AS_WE	1		1/2
48	C	S_WE_	1,3	1/2	1/3'
49	A	_WE_C	1		1/2
50	N	WE_CA	1		1/2
Итого бит				62,93 +	185,85 = 248,78

Шаг 7. Контекстом длины 1 по отношению к C является пробел. После пробела встречалась только буква W. Нужно передать *esc* (его вероятность равна 1/2) и перейти к контексту нулевой длины. Но и при контексте нулевой длины буква C не встречалась. Снова надо передать *esc*, вероятность которого равна 1/7. Однако, известно, что среди возможных букв не может быть буквы W. Это повышает вероятность *esc*-символа до 1/6.

Шаг 22. Это наиболее типичный шаг, алгоритм работает на этом шаге весьма эффективно. Контекстом по отношению к пробелу выступает _WE . Эта комбинация уже встречалась, и за ней следовал пробел. Таким образом следующими буквами могут быть пробел и *esc* с равными вероятностями. Поэтому для передачи пробела потребуется всего 1 бит.

Шаг 23. Букве W предшествует контекст _WE_. При предыдущем появлении этого контекста за ним следовала другая буква (C). Поэтому передается *esc* (вероятность 1/2). Контекст укорачивается и преобразовывается в WE_. За этим контекстом ранее следовала только C и надо снова передавать *esc*. Его вероятность без учета исключений была бы равна 1/2. Но при передаче предыдущего *esc* мы уже информировали, что буква C исключена, поэтому вероятность *esc*-символа равна 1. Контекст сократился до E_. Снова вероятность *esc* равна 1, и контекст сокращается до пробела. После пробела встречались W(2 раза), C, D и A. Буква C невозможна, значит вероятность W равна 2/5.

Шаг 32. Контекст к S – последовательность"_WE_". Она встречалась дважды, но не разу после нее не было буквы S. Передаем *esc*, имеющий вероятность 1/3. Последовательно укорачивая контекст, приходим к контексту, состоящему из одного пробела. Пробел уже встречался 7 раз, но ни разу перед S. Исключаем буквы C и W, тогда возможны только D и A. Поэтому вероятность *esc* равна 1/3. При подсчете вероятности буквы S при пустом контексте из 31 возможности нужно исключить все буквы, которые следовали после пробела. Окончательный результат 1/25.

Аналогичные, но еще более запутанные подсчеты выполняет кодер для алгоритма PPMD. Как и должно было получиться, PPMD экономит биты на контекстах и больше бит тратит на передачу букв. Поскольку в нашем примере мы кодируем короткую последовательность букв, алгоритм PPMD оказался заметно эффективнее алгоритма PPMA. ■

Таблица 4.5.1

Пример использования алгоритма PPMD

Шаг	Буква	Контекст	$\tau_n(\mathbf{c})$	$M_n(\mathbf{c})$	PPMD	
					$\hat{p}(esc \mid \mathbf{c})$	$\hat{p}(a \mid \mathbf{c})$
1	I	#	0	0	1	1/256
2	F	#	1	1	1/2	1/255
3	_	#	2	2	2/4	1/254
4	W	#	3	3	3/6	1/253
5	E	#	4	4	4/8	1/252
6	_	#	5	5		1/10
7	C	_	1,6	1,5	1/2×5/8'	1/251
8	A	#	7	6	6/14	1/250
9	N	#	8	7	7/16	1/249
10	N	#	9	8		1/18
11	O	N	1,10	1,8	1/2×7/16'	1/248
12	T	#	11	9	9/22	1/247
13	_	#	12	10		3/24
14	D	_	2,13	2,10	2/4×8/22'	1/246
15	O	#	14	11		1/28
16	_	O	1,15	1,11	1/2	5/28'
17	A	_	3,16	3,11	3/6	1/26'
18	S	A	1,17	1,11	1/2×11/34	1/245
19	_	#	18	12		7/36
20	W	_	4	4		1/8
21	E	_W	1	1		1/2
22	_	_WE	1	1		1/2
23	W	_WE_	1,1,1,5	1,1,1,4	1/2×1'×1'	3/8'
24	O	_W	2,2,23	1,1,12	2/4×1'	3/42
25	U	O	2,24	2,12		1/244
26	L	#	25	13	13/50	1/243
27	D	#	26	14		1/52
28	_	D	1,27	1,14	1/2	11/52'
29	W	_	6	4		5/12
30	E	_W	3	2		3/6
31	_	_WE	2	1		3/4
32	S	_WE_	2,2,2,7,31	2,2,2,4,14	2/4×1'×1'×2/4'	1/48'
33	H	S	1,32	1,14	1/2×13/50	1/242
34	O	#	33	15		5/66
35	U	O	3	3		1/6
36	L	OU	1	1		1/2
37	D	OUL	1	1		1/2
38	_	OULD	1	1		1/2
39	D	OULD_	1,1,1,1,8	1,1,1,1,5	1/2×1'×1'×1'	1/8
40	O	_D	1	1		1/2
41	_	_DO	1	1		1/2
42	A	_DO_	1	1		1/2
43	S	_DO_A	1	1		1/2
44	_	DO_AS	1	1		1/2
45	W	O_AS_	1	1		1/2
46	E	_AS_W	1	1		1/2
47	_	AS_WE	1	1		1/2
48	C	S_WE_	1,3	1,3	1/2	1/4
49	A	_WE_C	1	1		1/2
50	N	WE_CA	1	1		1/2
Итого					30,55 +	194,98 = 225,53

4.6. Сжатие с использованием преобразования Барроуза-Уилера

Описываемый в данном параграфе способ кодирования значительно отличается от всех рассмотренных ранее. Авторы этого способа, Барроуз и Уилер, описали его впервые в техническом отчете [BW94] в 1994 г. Предложенный алгоритм показывал неожиданно хорошие результаты, но далеко не рекордные в смысле соотношения между сложностью реализации и достижимым сжатием. С тех пор значительные усилия специалистов в области сжатия данных были направлены на то, чтобы повысить эффективность кодирования и найти теоретическое обоснование алгоритма. В результате на основе преобразования Барроуза-Уилера были построены архиваторы, которые наравне с PPM-кодерами являются рекордсменами по сжатию при сравнении на стандартных наборах файлов (см. раздел 4.7).

Опишем сначала *прямое преобразование Барроуза-Уиллера*. Будем иллюстрировать его на примере последовательности

IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN

Все циклические сдвиги последовательности должны быть лексикографически упорядочены. Результат выполнения этого шага для данного примера представлен на рис 3.x. Результатом преобразования являются

- последний столбец таблицы, строками которой служат лексикографически упорядоченные циклические сдвиги исходной последовательности;
- номер той строки таблицы, последним элементом которой является первая буква исходной последовательности.

В нашем примере результатами преобразования являются последовательность

CC_____LLWWWWISNUUAANNHWDD_AAOOO_____OOEEDTESFDSE

и число 14.

Отметим примечательные свойства преобразованной последовательности. Символы в этой ней «сгруппировались». Последовательность почти целиком состоит из серий одинаковых символов. Это дает основание надеяться на то, что сжимать эту последовательность будет проще, чем исходную. Платой является необходимость передать номер позиции, на которой находится первый символ исходной последовательности.

Покажем, что по результатам преобразования можно восстановить исходную последовательность. Эта задача решается с помощью *обратного преобразования* `Барроуза-Уиллера`. Начнем с примера.

В нашем примере входной информацией для обратного преобразования является последовательность

CC_____LLWWWWISNUUAANNHWDD_AAOOO_____OOEEDTESFDSE

и индекс первого символа 14. Очевидно, мы легко можем восстановить первый столбец таблицы циклических сдвигов. Для этого достаточно просто произвести сортировку преобразованной последовательности. Итак, в таблице на рис. 3.x. мы знаем первый, последний столбец и индекс первого символа I (число 14). В силу цикличности первый символ каждой строки следовал в исходной последовательности за последним символом той же строки. Отсюда заключаем, что в нашем примере, вторым символом был F. Каким был следующий символ? Находим F в последнем столбце (строка 46) и считываем первый символ той же строки. Оказывается, третьим символом был пробел.

0	A	NIF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_	C
1	A	NNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CANIF_WE_	C
2	A	S_WE_CANIF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO	_
3	A	S_WE_WOULD_WE_SHOULD_DO_AS_WE_CANIF_WE_CANNOT_DO	_
4	C	ANIF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE	_
5	C	ANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CANIF_WE	_
6	D	O_AS_WE_CANIF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD	_
7	D	O_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CANIF_WE_CANNOT	_
8	D	_DO_AS_WE_CANIF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOU	L
9	D	_WE_SHOULD_DO_AS_WE_CANIF_WE_CANNOT DO_AS_WE_WOU	L
10	E	_CAN IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_	W
11	E	_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CANIF_	W
12	E	_SHOULD_DO_AS_WE_CAN IF_WE_CANNOT_DO_AS_WE_WOULD_	W
13	E	_WOULD_WE_SHOULD_DO_AS_WE_CAN IF_WE_CANNOT_DO_AS_	W
14	F	_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN	I*
15	H	OULD_DO_AS_WE_CANIF_WE_CANNOT_DO_AS_WE_WOULD_WE_	S
16	I	F_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CA	N
17	L	D_DO_AS_WE_CANIF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHO	U
18	L	D_WE_SHOULD_DO_AS_WE_CANIF_WE_CANNOT_DO_AS_WE_WO	U
19	N	IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_C	A
20	N	NOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CANIF_WE_C	A
21	N	OT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CANIF_WE_CA	N
22	O	T_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CANIF_WE_CAN	N
23	O	ULD_DO_AS_WE_CANIF_WE_CANNOT_DO_AS_WE_WOULD_WE_S	H
24	O	ULD_WE_SHOULD_DO_AS_WE_CAN IF_WE_CANNOT_DO_AS_WE_	W
25	O	_AS_WE_CAN IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_	D
26	O	_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN IF_WE_CANNOT_	D
27	S	HOULD_DO_AS_WE_CANIF_WE_CANNOT_DO_AS_WE_WOULD_WE	_
28	S	_WE_CAN IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_	A
29	S	_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN IF_WE_CANNOT_DO_	A
30	T	_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CANIF_WE_CANN	O
31	U	LD_DO_AS_WE_CANIF_WE_CANNOT_DO_AS_WE_WOULD_WE_SH	O
32	U	LD_WE_SHOULD_DO_AS_WE_CANIF_WE_CANNOT_DO_AS_WE_W	O
33	W	E_CANIF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS	_
34	W	E_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CANIF	_
35	W	E_SHOULD_DO_AS_WE_CAN IF_WE_CANNOT_DO_AS_WE_WOULD	_
36	W	E_WOULD_WE_SHOULD_DO_AS_WE_CAN IF_WE_CANNOT_DO_AS	_
37	W	OULD_WE_SHOULD_DO_AS_WE_CAN IF_WE_CANNOT_DO_AS_WE	_
38	_	AS_WE_CANIF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_D	O
38	_	AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CANIF_WE_CANNOT_D	O
40	_	CANIF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_W	E
41	_	CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CANIF_W	E
42	_	DO_AS_WE_CANIF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOUL	D
43	_	DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CANIF_WE_CANNO	T
44	_	SHOULD_DO_AS_WE_CANIF_WE_CANNOT_DO_AS_WE_WOULD_W	E
45	_	WE_CANIF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_A	S
46	_	WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CANI	F
47	_	WE_SHOULD_DO_AS_WE_CANIF_WE_CANNOT_DO_AS_WE_WOUL	D
48	_	WE_WOULD_WE_SHOULD_DO_AS_WE_CANIF_WE_CANNOT_DO_A	S
49	_	WOULD_WE_SHOULD_DO_AS_WE_CANIF_WE_CANNOT_DO_AS_W	E

Рис.4.6.1. Лексикографическая сортировка циклических сдвигов

В этом месте мы сталкиваемся с небольшой проблемой. Пробелов в последнем столбце много. Какой из них «правильный»? Выручает то, что все строки таблицы лексикографически упорядочены. Одинаковые символы последнего столбца упорядочены по значениям соответствующих символов первого столбца. Букве F последнего столбца соответствует 9-й по счету пробел первого столбца. Значит, чтобы найти «наш» пробел, мы должны пропустить 8 пробелов последнего столбца и остановиться на девятом. Таким образом мы попадаем на строку 34. В ее первом столбце находим следующий символ W. Продолжая, мы правильно восстановим всю последовательность.

Опишем алгоритм формально. Декодер строит так называемый *вектор преобразования* T, элемент которого T[i]– номер той позиции преобразованной последовательности, на которой расположен (i+1)-й символ исходной последовательности. Фрагмент С-кода обратного преобразования приведен на рис. 3.xx. Промежуточные результаты вычислений представлены в таблице 3.x .

```
// Построение композиции Count
for ( i = 0 ; i < 257 ; i++ ) Count[ i ] = 0;
for ( i = 0 ; i < length ; i++ ) Count[ buffer[ i ] ]++;
// Построение кумулятивной композиции RunningTotal
sum = 0;
for ( i = 0 ; i < 257 ; i++ )
{
   RunningTotal[ i ] = sum;
   sum += Count[ i ];
   Count[ i ] = 0;
}
// Вычисление вектора преобразования T
for ( i = 0 ; i < length ; i++ )
{
   index = buffer[ i ];
   j= Count[ index ] + RunningTotal[ index ];
   T[ j ] = i;
   Count[ index ]++;
}
// Считывание данных в соответствии с вектором T
i = first;
for ( j = 0 ; j < length; j++ )
{
   out[j] = buffer[ i ];
   i = T[ i ];
}
```

Рис 4.6.2 Фрагмент С-кода обратного преобразования Барроуза-Уилера

Таблица 4.6.1

Предварительные вычисления при построении вектора преобразования

Буквы	Композиция Count	Кумулятивная композиция RunningTotal
A	4	0
C	2	4
D	4	6
E	4	10
F	1	14
H	1	15
I	1	16
L	2	17
N	3	19
O	5	22
S	3	27
T	1	30
U	2	31
W	5	33
_	12	38

Переменная length содержит длину входной последовательности декодера, сама последовательность находится в массиве buffer. Сначала вычисляется композиция Count и кумулятивная композиция RunningTotal. Затем по ним вычисляется вектор T, который в нашем примере имеет вид

T=[4,5,38,39,40,41,42,43,17,18,…].

Построение T показано в таблице 4.6.2. По вектору T легко восстанавливаем исходную последовательность:

i=first=14
Out[0]=buffer[14]=I;
i=T[14]=46;
Out[1]=buffer[46]=F;
i=T[46]=34;
Out[2]=buffer[34]=_;
i=T[34]=11;
Out[3]=buffer[11]=W;
i=T[11]=41;
Out[4]=buffer[41]=E;
i=T[41]=5;
и т.д.

Итак, мы убедились в том, что индекса первой буквы и преобразованной последовательности достаточно для восстановления исходной последовательности. Теперь нужно выбрать способ кодирования преобразованной последовательности. Для этого целесообразно использовать описанный выше метод стопки книг.

Таблица 4.6.2

Построение вектора обратного преобразования

I	0	1	2	3	4	5	6	7	8	9
buffer	C	C	_	_	_	_	_	_	L	L
j	4	5	38	39	40	41	42	43	17	18
T	19	20	28	29	0	1	25	26	42	47
I	10	11	12	13	14	15	16	17	18	19
buffer	W	W	W	W	I	S	N	U	U	A
j	33	34	35	36	16	27	19	31	32	0
T	40	41	44	49	46	33	4	8	9	16
I	20	21	22	23	24	25	26	27	28	29
buffer	A	N	N	H	W	D	D	_	A	A
j	1	20	21	15	37	6	7	44	2	3
T	21	22	30	31	32	38	39	55	58	63
i	30	31	32	33	34	35	36	37	38	39
buffer	O	O	O	_	_	_	_	_	O	O
j	22	23	24	45	46	47	48	49	25	26
T	43	17	18	10	11	12	13	24	25	36
i	40	41	42	43	44	45	46	47	48	49
buffer	E	E	D	T	E	S	F	D	S	E
j	10	11	8	30	12	28	14	9	29	13
T	4	5	6	7	27	33	34	35	36	37

4.7. Сравнение способов кодирования. Характеристики архиваторов

Раздел 5
Кодирование для каналов с шумом

В предыдущих разделах мы решали задачу кодирования источников. Цель кодирования состояла в уменьшении затрат на передачу либо хранение информации. Решение задачи может быть интерпретировано как «устранение» из потока передаваемых данных избыточной информации, без которой возможно однозначное восстановление сообщений источника.

В настоящем разделе рассматривается кодирование с целью защиты передаваемой информации от помех и искажений при передаче по каналам связи, либо при хранении информации. Решение задачи состоит в том, что при кодировании в информацию искусственно вносится избыточность таким образом, чтобы при искажении части передаваемых данных сообщения источника могли быть правильно декодированы. Таким образом, в некотором смысле задача кодирования для каналов противоположна задаче кодирования источника.

В повседневной жизни мы используем примерно такой же способ защиты информации от помех. Избыточность языка общения людей довольно высока (неслучайно тексты сжимаются программами-архиваторами почти в 5 раз). Эта избыточность позволяет хорошо понимать собеседника, даже если разговор происходит в шумном помещении, или по телефонному каналу с большим уровнем помех, или если дикция собеседника несовершенна.

Для построения эффективной системы связи теория информации рекомендует сначала удалить избыточность из сообщений для уменьшения объема передаваемой информации, а затем вводить избыточность для защиты информации от ошибок. Приведенный выше пример показывает, что возможен другой путь: можно использовать уже имеющуюся избыточность источника. Однако раздельное решение задачи кодирования источников и задачи кодирования для каналов оказывается более продуктивным и позволяет добиться впечатляющих результатов.

В процессе изучения кодирования источников мы описали конструктивные методы кодирования, реально используемые сегодня в практике сжимающего кодирования. К сожалению, в задаче кодирования для каналов связи мы не сумеем в рамках короткого курса пройти путь от постановки задачи до используемых на практике способов защиты информации от ошибок. Более того, теория кодов, исправляющих ошибки ─ тема самостоятельного курса, опирающегося на математический аппарат линейной алгебры, комбинаторики и теории конечных полей Галуа.

Раздел, посвященный каналам связи, начинается с постановки задачи кодирования для каналов. Далее рассматриваются модели каналов. Затем вводится понятие взаимной информации между случайными ансамблями, и для самых простых моделей устанавливается предел достижимой скорости кодирования (пропускная способность каналов). Завершается раздел доказательством прямой и обратной теорем кодирования.

5.1. Постановка задачи помехоустойчивого кодирования

Начнем с примера. Рассмотрим систему связи, в которой входом канала являются двоичные сигналы, соответствующие символам алфавита $X = \{0,1\}$. На выходе канала каждый сигнал анализируется и выносится решение в пользу одного из двоичных символов выходного алфавита $Y = X$.

Будем считать, что подлежащая передаче информация представлена в двоичной форме

и поэтому элементарные сообщения представляют собой также двоичные символы 0 либо 1. Таким образом, мы можем непосредственно использовать входные символы канала для передачи сообщений источника. Предположим, однако, что из-за шума в канале связи при передаче сигналов возможны ошибки. Как можно усовершенствовать систему связи, с тем, чтобы устранить ошибки или, по меньшей мере, уменьшить их долю в принятой последовательности сообщений?

Приведем примеры кодов, исправляющих ошибки.

Пример 1. Будем вместо сообщения 0 передавать последовательность 000, а вместо 1 – последовательность 111. Множество передаваемых по каналу последовательностей образует код, который в данном случае имеет длину 3 и содержит 2 кодовых слова (мощность кода равна 2). Считая, что правильная передача символов более вероятна, чем неправильная, легко предложить разумную стратегию принятия решений декодером. Если принятая из канала последовательность «больше похожа» на 000, чем на 111, декодер считает, что передавалось сообщение 0, в противном случае принимает решение в пользу сообщения 1. Иначе говоря, если число единиц меньше 2, решение принимается в пользу 0, в противном случае – в пользу 1.

Число несовпадающих позиций в двух последовательностях **x** и **y** называют *расстоянием Хэмминга* между этими последовательностями. Описанный в примере декодер является декодером, принимающим решения по принципу минимума расстояния Хэмминга.

Назовем множество последовательностей **y** на выходе канала, декодируемых в пользу некоторого кодового слова (точнее, в пользу сообщения, соответствующего этому слову), *решающей областью* слова (сообщения). Решающие области для данного примера приведены в таблице 5.1.

Таблица 5.1.

Корректирующий код из примера 5.1.

Сообщения	Кодовые слова	Решающие области
0	000	{000, 001, 010, 100}
1	111	{011, 101, 110, 111}

Нетрудно видеть, что одиночная ошибка в канале связи при использовании такого кода не опасна. Декодер примет правильное решение. Про такой код говорят, что он исправляет все ошибки кратности 1. Заметим, что на передачу одного бита информации данный код затрачивает 3 двоичных сигнала. Разумной характеристикой кода служит его скорость, которая указывает количество бит информации, переносимых одним сигналом. Скорость кода в данном примере равна

$$R = 1/3 = 0{,}33 \text{ (бит/символ канала).} \blacksquare$$

Рассмотрим еще один, немного более сложный, пример кода, исправляющего одиночные ошибки.

Пример 5.2. Объединим биты передаваемого сообщения в пары. Множество «расширенных сообщений» состоит теперь из 4 различных двоичных комбинаций. Приведенный в таблице 5.2. пример показывает одну из возможных конструкций кода для этого множества сообщений.

Таблица 5.2.

Код для примера 5.2.

Сообщения	Кодовые слова	Решающие области
00	00000	{00000,00001,00010,00100,01000,10000,11000,10001}
01	10110	{10110,10111,10100,10010,11110,00110,01110,00111}
10	01011	{01011,01010,01001,01111,00011,11011,10011,11010}
11	11101	{11101,11100,11111,11001,10101,01101,00101,01100}

Непосредственной проверкой можно убедиться в том, что этот код также не боится однократных ошибок, но его скорость несколько больше, поскольку 5 сигналов будет затрачено уже на передачу 2 двоичных сообщений. Поэтому

$$R = 2/5 = 0{,}4 \text{ (бит/символ канала).} \blacksquare$$

Понятно, что рассмотренные коды не спасают полностью от ошибок при передаче сообщений. Код примера 5.1.1. не защищает от двукратных ошибок. Код примера 5.1.2 исправляет некоторые комбинации 2-кратных ошибок (например, 11000, 10001), но все остальные комбинации из 2 или большего числа ошибок неминуемо приводят к выдаче получателю неправильно восстановленных сообщений.

Приведенные выше примеры убеждают в том, что важными характеристиками кода, защищающего информацию от ошибок, являются его скорость и помехоустойчивость. Последняя характеристика может быть численно измерена вероятностью выдачи получателю ошибочного сообщения.

Перейдем теперь к формальной постановке задачи. Для этого рассмотрим схему, представленную на рис. 5.1.1.

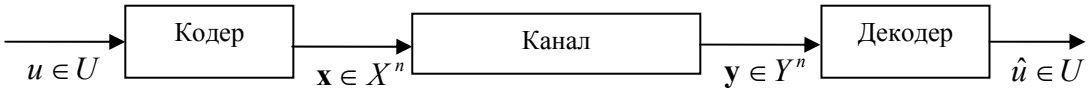


Рис. 5.1.1. Схема системы связи

В качестве множества сообщений без потери общности можно рассматривать множество чисел $U = \{1,...,M\}$.

Кодом канала называется над алфавитом любое множество последовательностей $A = \{\mathbf{x}_m\}$, $m = 1,...,M$, $A \in X^n$. Сами последовательности называются *кодовыми словами*, их длина n называется *длиной кода*, количество последовательностей M называется *мощностью кода,* величина

$$R = \frac{\log M}{n} \qquad (5.1.1)$$

называется скоростью кода. Скорость кода измеряется в битах на символ канала.

В частном случае когда $M = 2^k$, каждому из кодовых слов можно сопоставить последовательность из k информационных символов (k бит). Для их передачи будет использовано кодовое слово длины n. Скорость передачи в битах на символ канала составит $R = \log M / n = k / n$ бит/символ.

На каждом такте работы системы связи кодер получает от источника некоторое сообщение u и преобразует в соответствующее кодовое слово $\mathbf{x}$. В результате передачи по каналу на выходе канала появляется последовательность $\mathbf{y}$. Декодер по последовательности $\mathbf{y}$ вычисляет оценку номера переданного сообщения $\hat{u}$. Событие $\hat{u} \neq u$ называется ошибкой декодирования, а его вероятность – вероятностью ошибки.

Задача состоит в том, чтобы обеспечить наибольшую возможную скорость кода при наименьшей вероятности ошибки. Эти два требования противоречат друг другу. Приведенные выше примеры кодов показывают, что с увеличением числа кодовых слов уменьшаются размеры решающих областей, уменьшается расстояние между кодовыми словами, увеличивается вероятность ошибки. Тем не менее, оказывается, что для каждого канала можно указать скорость кода, при которой вероятность ошибки может быть сделана сколь угодно малой.

Число C называется *пропускной способностью канала*, если при любой скорости кода $R < C$ существуют коды, обеспечивающие сколь угодно малую вероятность ошибки и, напротив, при $R > C$ существует константа $\varepsilon > 0$, такая, что вероятность ошибки любого кода ограничена снизу величиной ε (т.е. вероятность ошибки не может быть сделана сколь угодно малой).

Для того, чтобы утверждать, что число C для заданной модели канала является пропускной способностью, нужно доказать две теоремы кодирования: прямую и обратную. Прямая теорема кодирования утверждает, что при любой скорости меньшей C достижима сколь угодно малая вероятность ошибки, обратная теорема – напротив, что при скорости большей C вероятность ошибки будет большой для любого кода.

При решении практических задач помехоустойчивого кодирования помимо скорости и вероятности ошибки необходимо принимать во внимание еще один параметр системы связи – сложность ее практической реализации. Как уже говорилось, практические аспекты построения корректирующих кодов не рассматриваются в данном курсе. В последующих параграфах мы научимся вычислять пропускную способность для некоторых простых моделей каналов и докажем соответствующие теоремы кодирования.

5.2. Модели каналов

У нас есть опыт построения математической модели источника. Напомним, что для описания источника мы требовали указания всех многомерных распределений вероятностей на буквах источника.

Как мы уже знаем, канал преобразует дискретную входную последовательность над алфавитом X в выходную последовательность, элементы которой выбираются из, вообще говоря, другого алфавита Y. В реальных каналах искажения сигнала в процессе передачи имеют разнообразную природу, например, одним из источников помех является тепловой шум в среде распространения сигналов. В любом случае нам удобно предположить, что искажения носят случайный характер. Это предположение позволяет описывать канал, задавая условные распределения вероятностей выходных последовательностей при известных последовательностях на входе канала.

Входными воздействиями реального физического канала связи обычно служат электрические сигналы, являющиеся непрерывными функциями времени, множество их значений также непрерывно. Множество значений сигналов на входе модулятора, также как и множество решений, принимаемых демодулятором, дискретно. В зависимости от того, какую часть системы связи мы объединяем в понятии «канал», можно рассматривать каналы непрерывного времени и дискретного времени, каналы с дискретным и (или) непрерывным входом и выходом. Из всего множества ситуаций мы рассмотрим только дискретные (по множеству входных и выходных символов) каналы дискретного времени.

Мы будем говорить, что модель канала задана, если для любых n и любых последовательностей $\mathbf{x} \in X^n$, $\mathbf{y} \in Y^n$ указано правило вычисления условной вероятности $p(\mathbf{y} \mid \mathbf{x})$.

Напомним обозначение $\mathbf{x}_i^n = (x_i, ..., x_n)$. Канал называется *стационарным*, если для любых j и n и любых $\mathbf{x}_{j+1}^{j+n} \in X^n$, $\mathbf{y}_{j+1}^{j+n} \in Y^n$ условные вероятности $p(\mathbf{y}_{j+1}^{j+n} \mid \mathbf{x}_{j+1}^{j+n})$ не зависят определяются однозначно значениями символов последовательностей и не зависят

от положения последовательностей во времени.

Канал называется *каналом без памяти*, если для любых j и n и любых $\mathbf{x}_{j+1}^{j+n} \in X^n$, $\mathbf{y}_{j+1}^{j+n} \in Y^n$

$$p(\mathbf{y}_{j+1}^{j+n} \mid \mathbf{x}_{j+1}^{j+n}) = \prod_{i=j+1}^{j+n} p(y_i \mid x_i).$$

Согласно данному определению, в канале без памяти события, происходящие при передаче последовательных символов, независимы.

Стационарный канал без памяти называют *дискретным постоянным каналом* (ДПК).

Заметим, что классификация каналов аналогична классификации источников. Однако задача кодирования для каналов значительно сложнее. Для произвольного стационарного источника мы легко доказали прямую и обратную теоремы кодирования и указали конструктивные асимптотически оптимальные методы кодирования. Больше того, оптимальное кодирование оказалось возможным даже при отсутствии априорной информации о модели источника. Среди всех каналов связи основным объектом нашего внимания будет ДПК.

Для описания ДПК достаточно указать одномерные условные вероятности $\{p(y \mid x), x \in X, y \in Y\}$. Положим $X = \{0,...,K-1\}$, $Y = \{0,...,L-1\}$. Обозначим $p_{ij} = p(y = j \mid x = i\}$. Переходные вероятности канала p_{ij} удобно записывать в виде матрицы

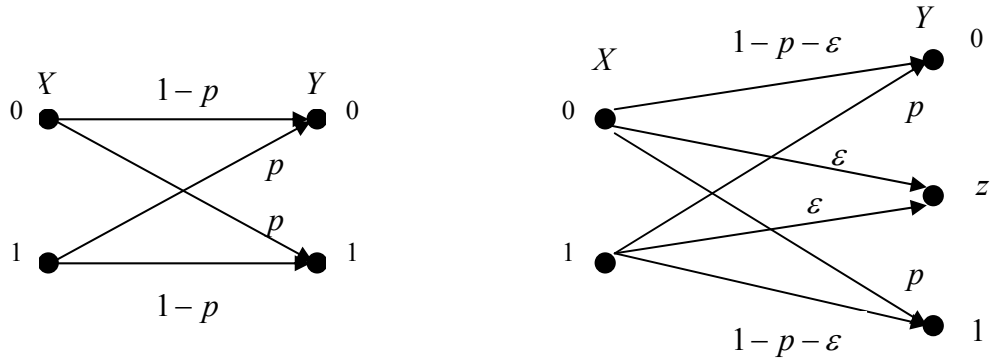
$$P = \begin{Vmatrix} p_{00} & \cdots & p_{0,L-1} \\ \cdots & \cdots & \cdots \\ p_{K-1,0} & \cdots & p_{K-1,L-1} \end{Vmatrix}.$$

Эта стохастическая матрица называется матрицей переходных вероятностей канала. Она полностью описывает модель ДПК. Приведем два простых, но важных примера ДПК.

Пример 5.2.1. Двоичный симметричный канал (ДСК). Для данной модели $X = Y = \{0,1\}$, а переходные вероятности удовлетворяют условиям $p_{10} = p_{01} = p$, $p_{00} = p_{11} = 1 - p$. Матрица переходных вероятностей имеет вид

$$P = \begin{Vmatrix} 1-p & p \\ p & 1-p \end{Vmatrix}.$$

Удобной графической иллюстрацией ДПК являются диаграммы, пример которой для случая ДСК показан на рис. 5.2.1а. Узлы графа соответствуют входным и выходным символам, ребра показывают возможные трансформации символов в канале, над ребрами записаны соответствующие вероятности переходов.



а) ДСК б) ДСтК

Рис. 5.2.1. Примеры дискретных каналов.

В ДСК искажения последовательно передаваемых двоичных символов происходят независимо от результата передачи других символов, вероятность искажения p не изменяется во времени и не зависит от значения передаваемого символа. ■

Пример 5.2.2. Двоичный симметричный канал со стираниями (ДСтК). Диаграмма этой модели показана на рис. 5.2.1б, а матрица переходных вероятностей имеет вид

$$P = \begin{Vmatrix} 1-p-\varepsilon & \varepsilon & p \\ p & \varepsilon & 1-p-\varepsilon \end{Vmatrix}.$$

Входной алфавит ДСтК содержит два символа, 0 и 1, а выходной алфавит помимо этих двух символов содержит дополнительный символ z, который называют символом стирания. В ДСтК каждый из входных символов может с вероятностью p превратиться в противоположный символ и с вероятностью ε может принять неопределенное значение, называемое стиранием. Вероятности p и ε постоянны, они не зависят от результатов передачи предыдущих и следующих символов и от значения передаваемого в данный момент символа. ■

Эти две модели имеет следующую практическую интерпретацию. Рассмотрим работу достаточно типичной системы связи, в которой информация передается двоичными сигналами s_0 и s_1. Приемное устройство (демодулятор) анализирует выход канала в течение промежутка времени, соответствующего длительности элементарного сигнала и вычисляет некоторую скалярную величину μ. Решение принимается сравнением μ с некоторым порогом T. При $\mu > T$ принимается решение в пользу одного из двух сигналов, для определенности, s_1, в противном случае, при $\mu \le T$, решением будет s_0. При правильном выборе порога T вероятности ошибок при передаче сигналов будут одинаковыми, и мы приходим к модели ДСК. Недостаток такой простейшей схемы приема состоит в том, что демодулятор теряет информацию о надежности принимаемых сигналов. Очевидно, значениям $\mu \approx T$ соответствуют ненадежные решения, и эти сведения могут быть полезными при последующей обработке информации.

Систему можно усовершенствовать, введя «защитный интервал» или «зону стирания». При $\mu > T + \Delta$ и при $\mu < T - \Delta$ принимаются решения в пользу 0 и 1 соответственно, а при $T - \Delta \le \mu \le T + \Delta$ символ «стирается». Получаем модель ДСтК. Это небольшое изменение заметно повышает эффективность системы, поскольку задача исправления стираний проще задачи исправления ошибок. Один и тот же корректирующий код позволяет исправить примерно в 2 раза больше стираний, чем ошибок.

Добавим, что еще более эффективной (но и более сложной) будет система связи, в которой демодулятор не принимает решение о каждом отдельном символе. Выходом демодулятора является сама непрерывная случайная величина μ. Канал с дискретным входом и непрерывным выходом называют полунепрерывным каналом или каналом с мягкими решениями. Рассмотренные выше ДСК с ДСтК – каналы с жесткими решениями.

5.3. Взаимная информация. Средняя взаимная информация

При изучении каналов связи нас будет интересовать возможность получения информации о передаваемых сообщениях по символам, наблюдаемым на входе канала. Говоря более формально, для заданного произведения $XY = \{(x, y), p(x, y)\}$ дискретных ансамблей X и Y нужно количественно измерить информацию об элементах $x \in X$ входного ансамбля, содержащуюся в выходных символах $y \in Y$.

Подходящей мерой такой информации является *взаимная информация*, определяемая для любых пар $(x, y) \in XY$ соотношением

$$I(x; y) = I(x) - I(x \mid y). \qquad (5.3.1)$$

Отметим, что мы используем в выражении $I(x; y)$ в качестве разделителя точку с запятой, чтобы не перепутать эту величину с собственной информацией $I(x, y)$ пары сообщений (x, y).

Взаимная информация определена только для пар (x, y), имеющих ненулевую вероятность.

Попробуем пояснить смысл введенного понятия. Уменьшаемое $I(x)$ представляет собой количество собственной информации, содержащейся в сообщении x. Вычитаемое – условная собственная информация при известном y, иными словами, это количество информации, оставшейся в x после получения y. Таким образом, разность $I(x; y)$ представляет собой изменение информации в x благодаря получению y. Вполне разумно принять эту величину в качестве меры информации, содержащейся в y об x.

Отметим свойства взаимной информации.

1. Симметричность: $I(x; y) = I(y; x)$.
2. Если x и y независимы, то $I(x, y) = 0$.

Доказательства этих свойств следуют из определения взаимной информации и определений условной вероятности и независимости.

Взаимная информация характеризует пары сообщений и является случайной величиной определенной на произведении ансамблей XY.

Средней взаимной информацией между ансамблями X и Y называется величина

$$I(X; Y) = \mathbf{M}\big[I(x; y)\big].$$

Формула, выражающая среднюю взаимную информацию через совместное распределение вероятностей, имеет вид

$$I(X; Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{p(y \mid x)}{p(y)}. \qquad (5.3.2)$$

Заметим, что средняя взаимная информация определена для произвольных дискретных ансамблей, включая ансамбли, содержащие элементы, имеющие нулевую вероятность. (Вклад пар (x, y) таких, что $p(x, y) = 0$ в величину $I(X; Y)$ будет равен нулю).

Изучим свойства введенной информационной меры.

Свойство 5.3.1. Симметричность $I(X; Y) = I(Y; X)$**.**

Свойство 5.3.2. Неотрицательность: $I(X; Y) \geq 0$.

Свойство 5.3.3. Тождество $I(X; Y) = 0$. Имеет место тогда и только тогда, когда ансамбли X и Y независимы.

Свойство 5.3.4.

$$I(X; Y) = H(X) - H(X \mid Y) =$$
$$= H(Y) - H(Y \mid X) =$$
$$= H(X) + H(Y) - H(XY).$$

Свойство 5.3.5.

$$I(X; Y) \leq \min\{H(X), H(Y)\}.$$

Свойство 5.3.6.

$$I(X; Y) \leq \min\{\log |X|, \log |Y|\}.$$

Свойство 5.3.7. Взаимная информация $I(X; Y)$ – выпуклая $\cap$ функция распределения вероятностей $p(x)$.

Свойство 5.3.8 Взаимная информация $I(X;Y)$ – выпуклая $\cup$ функция условных распределений $p(y \mid x)$.

Доказательства. Свойство 5.3.1 следует из симметричности взаимной информации. Усредняя правую и левую часть (5.3.1) по всем парам $(x,y) \in XY$, и учитывая симметричность взаимной информации, получаем первые два тождества свойства 5.3.4. Последнее тождество следует из свойства 1.5.3 условной энтропии. Поскольку условная энтропия не превышает безусловной, из свойства 5.3.4 вытекает свойство 5.3.2. Условная энтропия равна безусловной только для независимых ансамблей, поэтому справедливо и свойство 5.3.3. Свойство 5.3.5 также следует из 5.3.4 и свойства неотрицательности условной энтропии. Далее, свойство 5.3.6 является следствием 5.3.5 и свойства 1.3.2 энтропии. Доказательству свойств 5.3.6 и 5.3.7 о выпуклости взаимной информации мы посвятим отдельный параграф 5.4. ■

Обсудим сформулированные и доказанные результаты. Мы интерпретируем среднюю взаимную информацию $I(X;Y)$ как среднее количество информации об элементах ансамбля X, содержащееся в элементах Y. Свойство 5.3.1 устанавливает, что $I(X;Y)$ – симметричная функция, характеризующая пару ансамблей (X,Y).

Применительно к каналам связи, можно говорить о том, что $I(X;Y)$ – это количество информации, переносимое одним символом канала. Свойство 5.3.2 говорит о том, что эта величина всегда неотрицательна, что вполне логично. При этом заметим, что взаимная информация $I(x;y)$ между отдельными элементами принимает как положительные, так и отрицательные значения.

Если выходные символы канала не зависят от входных, передача информации, разумеется, невозможна. Это обстоятельство отражено в свойстве 5.3.3. Свойства 5.3.4 – 5.3.6 полезны при вычислениях и для получения простых оценок взаимной информации.

Выпуклость функций используется в задачах оптимизации при поиске экстремальных значений. Переходные вероятности заданы изначально моделью канала, но входное распределение – во власти кодера. Свойство 5.3.7 по сути утверждает, что существует некоторое входное распределение, при котором взаимная информация максимальна. Это распределение оптимально в том смысле, что оно максимизирует количество информации, переносимое одним символом канала.

5.4. Условная средняя взаимная информация. Теорема о переработке информации

Введем еще одно понятие, используемое при анализе информационных характеристик систем. Рассмотрим произведение трех ансамблей $XYZ = \{(x,y,z), p(x,y,z)\}$. Зафиксируем элемент $z \in Z$ и рассмотрим условное распределение

$$p(x,y \mid z) = \frac{p(x,y,z)}{p(z)}.$$

Это распределение на ансамбле XY, равно как и любое другое, может быть использовано для вычисления средней взаимной информации между X и Y. Будем обозначать эту взаимную информацию как $I(X;Y \mid z)$. Формула для вычисления этой величины имеет вид

$$I(X;Y \mid z) = \sum_{x \in X} \sum_{y \in Y} p(x,y \mid z) \log \frac{p(y \mid x,z)}{p(y \mid z)}.$$

Эта величина принимает случайные значения, вероятности которых определяются распределением $p(z)$.

Средней условной взаимной информацией между X и Y при условии Z называется величина

$$I(X;Y \mid Z) = \mathbf{M}\big[I(X;Y \mid z\big] = \sum_{x\in X}\sum_{y\in Y}\sum_{z\in Z} p(x,y,z)\log\frac{p(y \mid x,z)}{p(y \mid z)}. \qquad (5.4.1)$$

Разумеется, средняя условная информация обладает всеми свойствами средней взаимной информации. Помимо этого, непосредственно из (5.4.1) получаем

$$I(X;Y \mid Z) = H(Y \mid Z) - H(Y \mid XZ). \qquad (5.4.2)$$

С помощью этой формулы легко доказать тождества

$$I(X;YZ) = I(X;Y) + I(X;Z \mid Y), \qquad (5.4.3)$$

$$I(X;YZ) = I(X;Z) + I(X;Y \mid Z). \qquad (5.4.4)$$

Первое из них доказывается следующей цепочкой преобразований:

$$\begin{aligned} &I(X;YZ) = H(X) - H(X \mid YZ) = \\ &= \big[H(X) - H(X \mid Y)\big] + \big[H(X \mid Y) - H(X \mid YZ)\big] = \\ &= I(X;Y) - I(X;Z \mid Y). \end{aligned}$$

Второе тождество доказывается аналогично.

Частный случай системы обработки информации, в которой мы имеем дело с тремя вероятностными ансамблями, показан на рис.5.4.1.

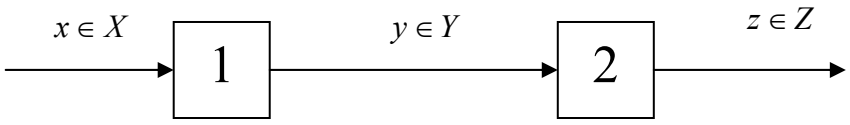


Рис.5.4.1. Система обработки информации

Входом системы являются элементы ансамбля X, выход первого устройства Y является входом второго, а выход второго устройства Z является выходом всей системы. Специфической особенностью распределения вероятностей на множестве XYZ является условная независимость X и Z при известном Y. Следствием этой независимости является теорема, называемая теоремой о переработке информации.

Теорема 5.4.1. Пусть X, Y и Z − вероятностные ансамбли, формируемые системой последовательной обработки информации, показанной на рис. 5.4.1. Тогда имеют места неравенства

$$I(X;Y) \ge I(X;Z), \qquad (5.4.5)$$

$$I(Y;Z) \ge I(X;Z). \qquad (5.4.6)$$

Доказательство. Воспользуемся (5.4.3) и (5.4.4):

$$I(X;YZ) = I(X;Y) + I(X;Z \mid Y), \qquad (5.4.7)$$

$$I(X;YZ) = I(X;Z) + I(X;Y \mid Z). \qquad (5.4.8)$$

В силу условной независимости X и Z при известном Y имеем $I(X;Z \mid Y) = 0$. Приравнивая правые части (5.4.7) и (5.4.8), получаем

$$I(X;Y) = I(X;Z) + I(X;Y \mid Z).$$

Поскольку второе слагаемое неотрицательно, получаем неравенство (5.4.5). Аналогично доказывается (5.4.6). ■

Неравенство (5.4.5) имеет следующее истолкование. Первое устройство можно рассматривать как канал связи, второе – как устройство обработки информации на выходе канала. Теорема утверждает, что при такой обработке взаимная информации между входом и выходом системы не увеличивается. Второе неравенство теоремы говорит о том, что обработка информации на входе системы также не приводит к увеличению информации между входом и выходом. Иными словами никакая обработка

информации (детерминированная или случайная) не позволяет увеличить количество информации о входе системы, получаемой на ее выходе.

Можно также рассматривать систему, показанную на рис. 5.4.1, как последовательное соединение двух каналов. В этом случае теорема устанавливает интуитивно понятный факт: количество, проходящей через последовательное соединение каналов, не может быть больше количества информации, проходящей через каждый отдельный канал.

5.5. Выпуклость средней взаимной информации.

В данном параграфе свойства средней условной взаимной информации будут использованы для доказательства выпуклости средней взаимной информации $I(X;Y)$ как функции входного распределения и переходных вероятностей.

Будем использовать вектор $\mathbf{p} = (p_0,...,p_{K-1})$ для обозначения распределения вероятностей на входных символах $X = \{0,...,K-1\}$. Для сокращения записи и чтобы подчеркнуть, что нас интересует зависимость взаимной информации от входного распределения, будем временно писать $I(\mathbf{p})$ вместо $I(X;Y)$.

В соответствии с определением выпуклой $\cap$ функции, нам предстоит доказать, состоит в том, что для любых двух стохастических векторов $\mathbf{p}_1$ и $\mathbf{p}_2$ и любого вещественного числа $\alpha \in [0,1]$ имеет место неравенство

$$I\left(\alpha\mathbf{p}_1 + (1-\alpha)\mathbf{p}_2\right) \geq \alpha I(\mathbf{p}_1) + (1-\alpha)I(\mathbf{p}_2). \qquad (5.5.1)$$

Введем в рассмотрение вспомогательное множество $Z = \{1,2\}$ и припишем его элементам вероятности $p_z(1) = \alpha$, $p_z(2) = 1-\alpha$.

Рассмотрим теперь произведение трех ансамблей XYZ, в котором тройки $(x,y,z) \in XYZ$ формируются по следующему правилу. Сначала в соответствии с распределением $p(z)$ выбирается один из двух элементов множества Z. Затем в соответствии с выбранным z выбирается одно из двух распределений $\mathbf{p}_1$ (если $z=1$) или $\mathbf{p}_2$ (если $z=2$) на множестве X. В соответствии с полученным распределением порождается элемент $x \in X$ и уже затем в соответствии с распределением $p(y \mid x)$ генерируется элемент $y \in Y$. Заметим, что ансамбли Z и Y условно независимы при условии X.

Согласно введенным обозначениям имеем

$$I(X;Y \mid z=1) = I(\mathbf{p}_1);$$
$$I(X;Y \mid z=2) = I(\mathbf{p}_2);$$
$$I(X;Y \mid Z) = \alpha I(\mathbf{p}_1) + (1-\alpha)I(\mathbf{p}_2);$$
$$I(X;Y) = I\left(\alpha\mathbf{p}_1 + (1-\alpha)\mathbf{p}_2\right).$$

Поэтому доказательство неравенства (5.5.1) сводится к доказательству неравенства

$$I(X;Y) \geq I(X;Y \mid Z). \qquad (5.5.2)$$

Чтобы доказать это утверждение, запишем двумя способами взаимную информацию $I(Y;XZ)$:

$$I(Y;XZ) = I(Y;X) + I(Y;Z \mid X); \qquad (5.5.3)$$
$$I(Y;XZ) = I(Y;Z) + I(Y;X \mid Z). \qquad (5.5.4)$$

В силу условной независимости Z и Y имеем $I(Y;Z \mid X) = 0$. Взаимная информация $I(Y;Z)$ неотрицательна. Поэтому, приравнивая правые части (5.5.3) и (5.5.4), получаем (5.5.2), что завершает доказательство выпуклости взаимной информации по входным распределениям.

Рассмотрим теперь взаимную информацию как функцию условных распределений $p(y \mid x)$. Будем временно обозначать через $I(P)$ среднюю взаимную информацию,

вычисленную для заданной стохастической матрицы $P = \{p(y \mid x), x \in X, y \in Y\}$. Требуется доказать, что для любых двух стохастических матриц P_1 и P_2 и любого $\alpha \in [0,1]$ имеет место неравенство

$$I\left(\alpha P_1 + (1-\alpha)P_2\right) \le \alpha I(P_1) + (1-\alpha) I(P_2). \qquad (5.5.5)$$

Снова рассмотрим вспомогательное множество $Z = \{1,2\}$ и припишем его элементам вероятности $p_z(1) = \alpha$, $p_z(2) = 1-\alpha$. Построим новое произведение трех ансамблей XYZ по следующему правилу. Сначала в соответствии с распределением $p(x)$ выбирается один из элементов множества X и независимо от него в соответствии с распределением $p(z)$ выбирается один из двух элементов множества Z Затем в соответствии с выбранным z выбирается одна из двух матриц переходных вероятностей $P = P_1$ (если $z = 1$) или $P = P_2$ (если $z = 2$). В соответствии с выбранными элементом x и матрицей P генерируется элемент $y \in Y$. Заметим, что в данной модели ансамбли Z и X независимы.

В соответствии с введенными обозначениями имеем

$$I(X;Y \mid z = 1) = I(P_1);$$
$$I(X;Y \mid z = 2) = I(P_2);$$
$$I(X;Y \mid Z) = \alpha I(P_1) + (1-\alpha) I(P_2);$$
$$I(X;Y) = I\left(\alpha P_1 + (1-\alpha)P_2\right).$$

Доказательство неравенства (5.5.5) теперь сводится к доказательству неравенства

$$I(X;Y) \le I(X;Y \mid Z). \qquad (5.5.6)$$

Запишем двумя способами взаимную информацию $I(X;YZ)$:

$$I(X;YZ) = I(X;Y) + I(X;Z \mid Y); \qquad (5.5.7)$$
$$I(X;YZ) = I(X;Z) + I(X;Y \mid Z). \qquad (5.5.8)$$

Из независимости X и Z имеем $I(X;Z) = 0$. Кроме того, $I(X;Z \mid Y) \ge 0$. Поэтому, приравнивая правые части (5.5.7) и (5.5.8), убеждаемся в справедливости (5.5.6) и, в свою очередь, (5.5.5). Тем самым доказана выпуклость $\cup$ взаимной информации по условным распределениям.

5.6. Информационная емкость и пропускная способность

Задача, которые мы решаем в данном параграфе, состоит в том, чтобы выразить достижимую скорость передачи информации по каналу через известные нам информационные меры, такие как энтропия, взаимная информация и т.п.

Рассмотрим дискретный стационарный канал. Мы знаем, что количество информации о входных символах X канала, содержащееся в выходных символах Y определяется средней взаимной информацией $I(X;Y)$. Это верно при передаче одного символа канала. Из опыта изучения источников информации известно, что кодирование последовательностей сообщений потенциально более эффективно, чем кодирование отдельных сообщений. Точно также при кодировании для каналов мы объединяем последовательные входные символы канала в блоки, представляющие собой кодовые слова некоторого кода.

Таким образом, при использовании кодов длины n количество информации, получаемой декодером при передаче одного кодового слова, в среднем составит $I(X^n;Y^n)$ бит, что соответствует скорости передачи информации

$$\frac{1}{n} I(X^n;Y^n) \text{ бит/символ канала.}$$

Эта величина зависит от переходных вероятностей $\{p(\mathbf{y} \mid \mathbf{x}),\ \mathbf{y} \in Y^n,\ \mathbf{x} \in X^n\}$ и от распределения вероятностей на входе канала $\{p(\mathbf{x}),\ \mathbf{x} \in X^n\}$. Входное распределение

следует выбрать таким, чтобы максимизировать скорость передачи информации. Результирующую скорость запишем в виде

$$\max_{\{p(\mathbf{x})\}} \frac{1}{n} I(X^n; Y^n) \text{ бит/символ канала.}$$

Заметим, что длина кода n также является свободным параметром, она может быть выбрана такой, чтобы скорость передачи была как можно больше. Эти неформальные рассуждения приводят нас к следующему определению.

Величина

$$C_0 = \sup_n \max_{\{p(\mathbf{x})\}} \frac{1}{n} I(X^n; Y^n) \tag{5.6.1}$$

называется *информационной емкостью* канала.

Напомним, что sup (supremum) обозначает наименьшую верхнюю грань множества, т.е. наименьшее число, не меньшее, чем любой из элементов множества. В отличие от максимального элемента, верхняя грань может не принадлежать множеству. Пусть, например $A = \{(n-1)/n, n = 1,2,...\}$. Максимального числа в этом множестве не существует, но $\sup A = 1$, причем эта верхняя грань достигается в пределе при $n \to \infty$. Аналогично в определении информационной емкости канала наибольшее значение средней взаимной информации на букву канала может достигаться при бесконечной длине кода, поэтому в (5.6.1) нас интересует именно верхняя грань, а не максимальное значение.

Итак, мы ввели в рассмотрение информационную емкость канала C_0 как некоторую функцию его переходных вероятностей. Мы предполагаем, что именно эта величина характеризует потенциально достижимую скорость передачи по каналу. Выше, в параграфе 5.1 мы определили пропускную способность канала C как максимальную скорость, при которой возможна передача со сколь угодно малой вероятностью ошибки. Естественно, возникает вопрос о том, как соотносятся между собой эти две характеристики канала.

В разделе 1, были введены два понятия – скорость создания информации источником H и предел энтропии на сообщение источника $H_\infty(X) = H(X \mid X^\infty)$. Было доказано, что для произвольного стационарного источника эти две величины равны.

Вопрос о соотношении меду информационной емкостью канала и его пропускной способностью несколько сложней. Ниже мы докажем обратную теорему кодирования, утверждающую, что информационная емкость C_0 ограничивает сверху скорость, при которой достижима сколь угодно малая вероятность ошибки. Для дискретного постоянного канала будет доказана прямая теорема кодирования, согласно которой при скорости сколь угодно близкой к C_0 достижима сколь угодно малая вероятность ошибки.

5.7. Неравенство Фано

Для доказательства теорем кодирования нам нужно будет установить связь между информационными характеристиками канала и источника с одной стороны и практически важными параметрами – скоростью передачи информации и вероятностью ошибки с другой стороны. Неравенство Фано – ключ к решению этой задачи.

На рис. 5.7.1 с целью иллюстрации вводимых ниже обозначений показана обобщенная схема системы передачи сообщений,

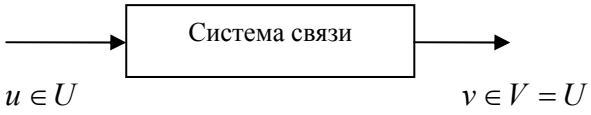


Рис. 5.7.1. Схема системы связи

Сообщениями являются элементы множества $U = \{u\} = \{0,...,M-1\}$. Получателю выдаются оценки сообщений. Множество оценок обозначено через $V = \{v\}$. Множества U и V совпадают. Всякий раз, когда $u \neq v$ имеет место ошибка декодирования.

Если задана модель системы связи, т.е. точно известны алгоритмы работы кодера и декодера и математическая модель канала, тем самым определено произведение ансамблей $UV = \{(u,v), p(u,v)\}$. С помощью распределения $p(u,v)$ вероятность ошибки P_e может быть вычислена по формуле

$$P_e = \sum_u \sum_{v \neq u} p(u,v). \tag{5.7.1}$$

Вероятность правильного решения равна

$$P_c = 1 - P_e = \sum_u \sum_{v=u} p(u,v). \tag{5.7.2}$$

Теорема 5.7.1. (Неравенство Фано)

$$H(U \mid V) \leq h(P_e) + P_e \log(M-1). \tag{5.7.3}$$

Обсуждение. Прежде чем мы приступим к доказательству этого неравенства, постараемся уяснить его смысл. В левой части неравенства имеем условную энтропию источника сообщений при условии, что известно вынесенное декодером решение. Эта величина характеризует неопределенность о переданном сообщении, оставшуюся на приемной стороне после вынесения решения. Каким образом можно было бы помочь декодеру устранить эту неопределенность? Предположим, что в распоряжении декодера имеется добрый, но практичный джин. Он знает истинное переданное сообщение, но требует плату за каждый бит полученной от него информации. Какие вопросы нужно задать джину?

Одна из нетривиальных стратегий – следующая. Сначала спросим джина, правильно ли принятое решение. Поскольку множество возможных ответов состоит из двух элементов, имеющих вероятности P_e и $P_c = 1 - P_e$, ответ джина обойдется нам в среднем в $h(P_e)$ бит. Если решение было правильным, (вероятность этого события $1 - P_e$), то дополнительной информации не требуется. Если же решение ошибочно, то достаточно узнать, какое из остальных $M-1$ возможных сообщений было передано. Эта вторая ситуация имеет вероятность P_e и обойдется нам в худшем случае в $\log(M-1)$ бит. Итого для разрешения неопределенности $H(U \mid V)$ нам понадобится не более

$$h(P_e) + (1 - P_e) \times 0 + P_e \log(M-1)$$

бит, что в точности совпадает с правой частью (5.7.3).

Эти эвристические рассуждения позволяют понять, почему неравенство Фано имеет место, и облегчают запоминание формулы в его правой части. Конечно, эти рассуждения нельзя считать доказательством неравенства Фано.

Обсудим коротко, какую полезную информацию мы можем извлечь из соотношения (5.7.3). Введем специальное обозначение $\gamma(P_e)$ для правой части неравенства Фано

$$\gamma(P_e) = h(P_e) + P_e \log(M-1) \tag{5.7.4}$$

и изучим поведение этой функции. В крайних точках 0 и 1 она равна соответственно 0 и $\log(M-1)$. Беря последовательно первую и вторую производную, убеждаемся в том, что функция $\gamma(P_e)$ строго выпукла $\cap$ и имеет максимум в точке $(M-1)/M$. В этой точке

$\gamma(P_e) = \log M$. График функции $\gamma(P_e)$ показан на рис. 5.7.2.

При доказательстве обратной теоремы кодирования нам понадобится следующее свойство функции $\gamma(P_e)$, которое поясняется представленным на рис. 5.7.2 графиком: для любого $\delta > 0$ существует положительное $\varepsilon > 0$ такое, что из неравенства $\gamma(P_e) > \delta$ следует неравенство $P_e > \varepsilon$.

Доказательство теоремы 5.7.1. Используя (5.7.1) и (5.7.2), запишем выражения, участвующие в неравенстве Фано (5.7.3), в следующем виде:

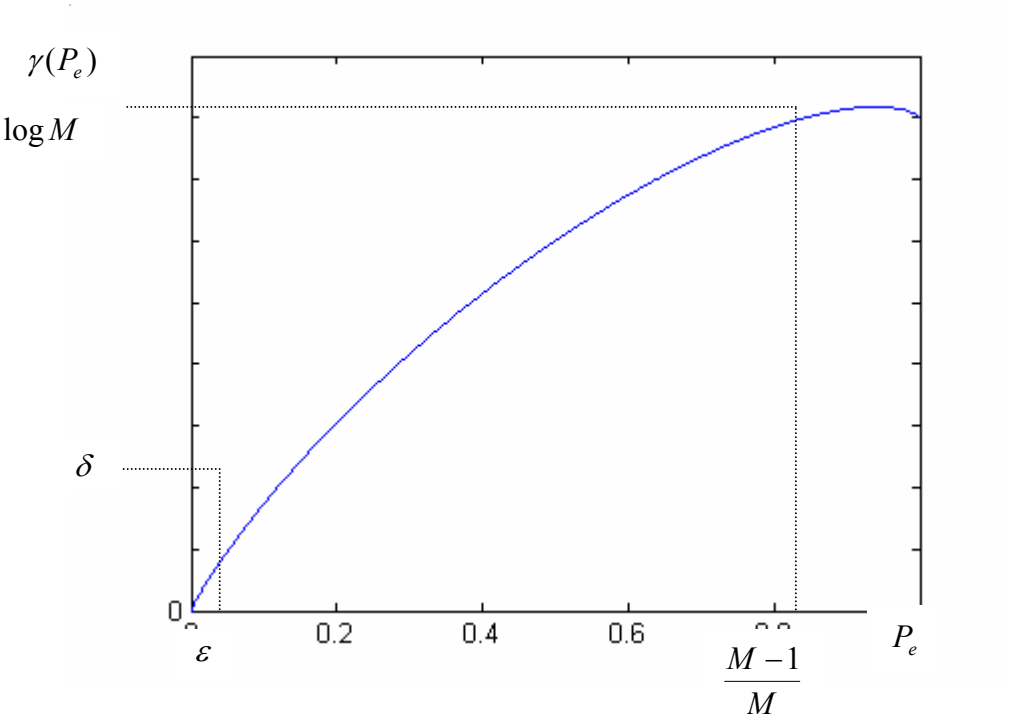


Рис. 5.7.2. График функции $\gamma(P_e)$

$$H(U \mid V) = -\sum_u \sum_{v \neq u} p(u,v) \log p(u \mid v) - \sum_u \sum_{v=u} p(u,v) \log p(u \mid v), \qquad (5.7.5)$$

$$h(P_e) = -\sum_u \sum_{v \neq u} p(u,v) \log P_e - \sum_u \sum_{v \neq u} p(u,v) \log P_c, \qquad (5.7.6)$$

$$P_e \log(M-1) = \sum_u \sum_{v \neq u} p(u,v) \log(M-1). \qquad (5.7.7)$$

Рассмотрим разность

$$\Delta = H(U \mid V) - h(P_e) - P_e \log(M-1).$$

Чтобы доказать (5.7.4), нужно доказать, что $\Delta \le 0$. Для этого вычтем из правой и левой части (5.7.5) соответствующие части тождеств (5.7.6) и (5.7.7) . После элементарных преобразований придем к равенству

$$\Delta = \sum_u \sum_{v \neq u} p(u,v) \log \frac{P_e}{p(u \mid v)(M-1)} + \sum_u \sum_{v=u} p(u,v) \log \frac{P_c}{p(u \mid v)}.$$

Следующий шаг основан на использовании неравенства

$$\log x \le (x-1)\log e.$$

Применив его к обоим слагаемым, получим

$$\Delta \le \log e \times$$

$$\times\left[\sum_u\sum_{v\neq u} p(u,v)\frac{P_e}{p(u\mid v)(M-1)} - \sum_u\sum_{v\neq u} p(u,v) + \sum_u\sum_{v=u} p(u,v)\frac{P_c}{p(u\mid v)} - \sum_u\sum_{v=u} p(u,v)\right].$$

Воспользуемся тем, что $p(u,v) = p(v)p(u\mid v)$ и обозначениями (5.7.1) и (5.7.2). Результат легко привести к виду

$$\Delta \le \log e \times\left[\frac{P_e}{M-1}\sum_u\sum_{v\neq u} p(v) - P_e + P_c\sum_u\sum_{v=u} p(v) - P_c\right]. \qquad (5.7.8)$$

Заметим теперь, что

$$\sum_u\sum_{v\neq u} p(v) = (M-1)\sum_v p(v) = (M-1). \qquad (5.7.9)$$

Здесь мы использовали то, что сумма в левой части содержит всего M различных слагаемых, и каждое из них встречается в этой сумме ровно $(M-1)$ раз. Кроме того, справедливо равенство

$$\sum_u\sum_{v=u} p(v) = \sum_u p(u) = 1. \qquad (5.7.10)$$

Подстановка (5.7.9) и (5.7.10) в (5.7.8) приводит к неравенству $\Delta \le 0$. Тем самым неравенство Фано доказано. ■

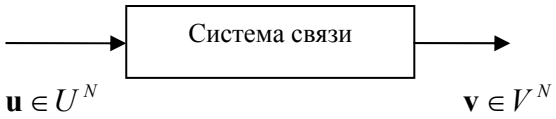


Рис. 5.7.3. Схема системы связи

Обобщим теорему 5.7.1 на случай передачи последовательностей сообщений. Для этого рассмотрим систему связи, показанную на рис. 5.7.3. На этот раз входом системы является последовательность сообщений $\mathbf{u} = (u_1,...,u_N)$, а выходом – последовательность решений $\mathbf{v} = (v_1,...,v_N)$, $u_i, v_i \in U = V = \{0,...,M-1\}$, $i = 1,...,N$. Вероятность ошибки при передаче i-го сообщения обозначается как $P_{ei} = P(u_i \neq v_i)$.

Введем в рассмотрение среднюю вероятность ошибки для последовательности длины N

$$\overline{P}_e = \frac{1}{N}\sum_{i=1}^{N} P_{ei}.$$

Теорема 5.7.2. Для последовательностей $(\mathbf{u},\mathbf{v}) \in U^N V^N$, составленных из элементов множества объема M, имеет место неравенство

$$\frac{1}{N}H(U^N \mid V^N) \le h(\overline{P}_e) + \overline{P}_e \log(M-1) \qquad (5.7.11)$$

Доказательство. Напомним, что условная энтропия не возрастает с увеличением числа условий. Поэтому

$$H(U^N \mid V^N) = \sum_{i=1}^{N} H(U_i \mid U_1...U_{i-1}V^N) \le \sum_{i=1}^{N} H(U_i \mid V_i).$$

Поделим обе части на N и к каждому слагаемому применим неравенство Фано. Получим

$$\frac{1}{N}H(U^N \mid V^N) \le \frac{1}{N}\sum_{i=1}^{N} h(P_{ei}) + \frac{1}{N}\sum_{i=1}^{N} P_{ei}\log(M-1).$$

Поскольку энтропия – выпуклая $\cap$ функция, средняя по i энтропия не меньше энтропии средней вероятности ошибки. Учитывая это, из последнего неравенства получаем доказываемое неравенство (5.7.11). ■

5.8. Обратная теорема кодирования

В этом параграфе для произвольного дискретного стационарного канала будет доказано, что надежная передача информации со скоростью, превышающей информационную емкость канала, невозможна. Прежде, чем будет точно сформулировано доказываемое утверждение, мы уточним, что в данном случае понимается под скоростью передачи информации.

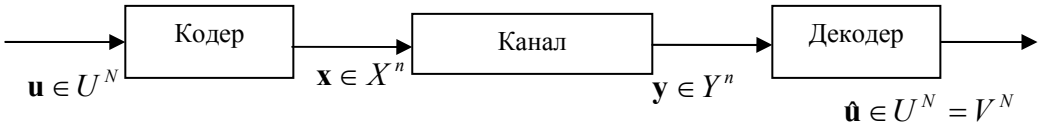


Рис. 5.8.1. Схема системы связи

Рассмотрим схему системы связи, представленную на рис. 5.8.1. Будем считать, что на вход канала символы канала $x \in X$ поступают через равные промежутки времени T_c, а источник информации порождает сообщения $u \in U$ через интервалы времени T_s. Введем обозначение $\lambda = T_c / T_s$.

Будем считать источник стационарным, и обозначим через H его скорость создания информации, т.е.

$$H = H_\infty(U) = H(U \mid U^\infty).$$

Для передачи N сообщений кодер может потратить не больше $\lfloor NT_s / T_c \rfloor = \lfloor N / \lambda \rfloor$ символов канала, где $\lfloor a \rfloor$ обозначает округление вниз до ближайшего целого. Хотя при некоторых значениях N число λn интервалов между сообщениями может не быть целым, в среднем за время передачи n символов канала от источника поступит $N = \lambda n$ сообщений, и поэтому скоростью передачи информации в данной системе связи мы считаем величину

$$R = \frac{N}{n} H \text{ бит/символ канала.}$$

Обозначим через P_{ei} вероятность несовпадения i-го сообщения входного блока $\mathbf{u} \in U^N$ с соответствующим сообщением в блоке $\hat{\mathbf{u}} \in U^N = V^N$ на выходе декодера, и через $\overline{P}_e = \frac{1}{N} \sum_{i=1}^{N} P_{ei}$ обозначим среднюю вероятность ошибки при передаче последовательности длины N.

Теорема 5.8.1. (Обратная теорема кодирования). Для дискретного стационарного канала с информационной емкостью C_0 для любого $\delta > 0$ при любой скорости передачи информации $R > C_0 + \delta$ существует число $\varepsilon > 0$, такое, что при использовании любого кода средняя вероятность ошибки удовлетворяет неравенству

$$\overline{P}_e \geq \varepsilon.$$

Доказательство. Дважды применив теорему о переработке информации **(**Теорема 5.4.1.), при любых N и n получим

$$I\left(U^N;V^N\right)\le I\left(X^n;Y^n\right).$$

Запишем левую часть в виде

$$I\left(U^N;V^N\right)=H\left(U^N\right)-H\left(U^N\mid V^N\right).$$

Получаем неравенство

$$H\left(U^N\mid V^N\right)\ge H\left(U^N\right)-I\left(X^n:Y^n\right).$$

Применив неравенство Фано (Теорема 5.7.2), получим

$$\gamma(\overline{P}_e)\ge\frac{1}{N}H\left(U^N\mid V^N\right)\ge\frac{1}{N}\left[H\left(U^N\right)-I\left(X^n;Y^n\right)\right], \qquad (5.8.1)$$

где функция $\gamma(\cdot)$ определена соотношением (5.7.4).

При любых N и n в силу свойств энтропии на сообщение стационарного источника и по определению информационной емкости справедливы неравенства

$$H\left(U^N\right)\ge NH\ ,\ \ I\left(X^n;Y^n\right)\le nC_0\,.$$

Их подстановка в (5.8.1) приводит к неравенству

$$\gamma(\overline{P}_e)\ge\frac{n}{N}\left[R-C_0\right]\ge\lambda\delta\,.$$

Из свойств функции $\gamma(\cdot)$ (см. обсуждение перед доказательством Теоремы 5.7.1) следует, что из неравенства $\gamma(\overline{P}_e)\ge\lambda\delta$ следует существование положительного $\varepsilon>0$ такого, что $\overline{P}_e\ge\varepsilon$. Тем самым теорема доказана. ■

Итак, доказанная теорема утверждает, что, если скорость передачи информации хотя бы ненамного (на любую положительную величину δ) превышает информационную емкость канала, вероятность ошибки уже не может быть сколь угодно малой. Она ограничена снизу положительной величиной ε, независящей от длины используемых кодов. Отметим особо, что результат получен для произвольного стационарного канала.

Чтобы доказать прямую теорему кодирования, нам придется существенно сузить множество рассматриваемых моделей. В следующих параграфах мы получим относительно простые формулы для информационной емкости некоторых каналов без памяти и уже затем докажем прямую теорему кодирования для дискретных постоянных каналов.

5.9. Вычисление информационной емкости каналов без памяти

Рассмотрим дискретный постоянный канал (стационарный канал без памяти), заданный входным и выходным алфавитами $X=\{x\}$ и $Y=\{y\}$ и матрицей условных распределений $P=\{p(y\mid x)\}$. Для данной модели канала для произвольных последовательностей $\mathbf{x}=(x_1,...,x_n)$ и $\mathbf{y}=(y_1,...,y_n)$ условные вероятности $p(\mathbf{y}\mid\mathbf{x})$ вычисляются по формуле

$$p(\mathbf{y}\mid\mathbf{x})=\prod_{i=1}^{n}p(y_i\mid x_i)\,. \qquad (5.9.1)$$

По определению, информационная емкость канала равна

$$C_0=\sup_n\max_{\{p(\mathbf{x})\}}\frac{1}{n}I(X^n;Y^n)\,. \qquad (5.9.2)$$

Эта формула непригодна для вычисления информационной емкости по заданной матрице P, поскольку предполагает вычисление экстремума функции по счетному числу параметров. Нам предстоит дать ответ на вопрос, нельзя ли упростить (5.9.2), используя (5.9.1)? Положительный ответ на этот вопрос сформулируем в виде следующей теоремы.

Теорема 5.9.1. Информационная емкость дискретного постоянного канала вычисляется по формуле

$$C_0=\max_{\{p(x)\}}I(X;Y)\,. \qquad (5.9.3)$$

Доказательство. Доказательство состоит из двух шагов. Сначала мы докажем, что правая часть (5.9.3) является границей сверху на информационную емкость канала. Затем мы докажем, что эта граница достигается при некотором распределении $p(\mathbf{x})$.

При произвольном распределении $\{p(\mathbf{x}), \mathbf{x} \in X^n\}$ запишем взаимную информацию между входными и выходными последовательностями в виде

$$I(X^n; Y^n) = H(Y^n) - H(Y^n \mid X^n). \tag{5.9.4}$$

Воспользовавшись (5.9.1) и свойствами логарифма и математического ожидания, выполним простые преобразования.

$$H(Y^n \mid X^n) = \mathbf{M}[-\log p(\mathbf{y} \mid \mathbf{x})] =$$

$$= \mathbf{M}\left[-\log \prod_{i=1}^{n} p(y_i|x_i)\right] = \sum_{i=1}^{n} \mathbf{M}[-\log p(y_i|x_i)] =$$

$$= \sum_{i=1}^{n} H(Y_i \mid X_i).$$

Из свойств энтропии мы помним, что

$$H(Y^n) \le \sum_{i=1}^{n} H(Y_i), \tag{5.9.5}$$

причем равенство имеет место только для независимых ансамблей. С учетом выполненных выкладок вместо (5.9.4) получаем неравенство

$$I(X^n; Y^n) \le \sum_{i=1}^{n} [H(Y_i) - H(Y_i \mid X_i)] = \sum_{i=1}^{n} I(X_i; Y_i). \tag{5.9.6}$$

Тем самым мы, по сути, завершили первый шаг нашего доказательства. Предположим теперь, что буквы на входе канала и докажем, что при этом предположении в (5.9.6) имеет мест равенство. Это будет означать, что максимум в (5.9.2) следует искать только среди таких распределений на X^n, которые порождают последовательности независимых букв на входе канала. Доказательство сводится к доказательству того, что при этом предположении равенство имеет место в (5.9.5). Проще говоря, нужно доказать, что для дискретного канала без памяти из независимости букв на входе канала следует независимость символов на выходе канала.

Выходное распределение по формуле полной вероятности вычисляется как

$$p(\mathbf{y}) = \sum_{\mathbf{x} \in X^n} p(\mathbf{x}) p(\mathbf{y} \mid \mathbf{x}).$$

В предположении о независимости входных символов с учетом (5.9.1) получаем

$$p(\mathbf{y}) = \sum_{\mathbf{x} \in X^n} \prod_{i=1}^{n} p(x_i) \prod_{i=1}^{n} p(y_i|x_i) =$$

$$= \sum_{\mathbf{x} \in X^n} \prod_{i=1}^{n} p(x_i) p(y_i \mid x_i) =$$

$$= \sum_{x_1 \in X} \sum_{x_2 \in X} \dots \sum_{x_n \in 2} [p(x_1) p(y_1 \mid x_1) \times p(x_2) p(y_2 \mid x_2) \times \dots \times p(x_n) p(y_n \mid x_n)].$$

Поочередно вынося сомножители за знаки сумм, придем к равенству

$$p(\mathbf{y}) = \prod_{i=1}^{n} \sum_{x_i \in X} p(x_i) p(y_i \mid x_i) = \prod_{i=1}^{n} p(y_i),$$

из которого и вытекает доказываемое утверждение о независимости выходных символов канала.

Теперь мы знаем, что при независимых входных символах в (5.9.6) имеет место равенство. Подстановка этого равенства в определение информационной емкости (5.9.2) дает

$$C_0 = \sup_n \max_{\{p(\mathbf{x})\}} \frac{1}{n} \sum_{i=1}^{n} I(X_i; Y_i) .$$

Заметим, что каждое слагаемое зависит только от распределения вероятностей для соответствующего входного символа. Следовательно, поиск максимума может быть выполнен для каждого отдельного слагаемого независимо от других. Получаем

$$C_0 = \sup_n \frac{1}{n} \sum_{i=1}^{n} \max_{\{p(x_i)\}} I(X_i; Y_i) .$$

В силу стационарности канала все слагаемые суммы будут одинаковыми. Поэтому

$$C_0 = \sup_n \max_{\{p(x)\}} I(X;Y) = \max_{\{p(x)\}} I(X;Y) . \blacksquare$$

Мы пришли к вполне естественному результату: для стационарного канала без памяти вычисление информационной емкости сводится к максимизации средней взаимной информации между отдельными буквами по одномерным распределениям.

Важен также доказанный попутно факт: пропускная способность канала без памяти достигается при независимых буквах на входе канала. В этом смысле распределения, порождающие независимые буквы на входе канала, являются оптимальными.

Формула (5.9.3) много проще исходной формулы (5.9.2). При небольшом объеме алфавита оптимизация по входным распределениям может быть выполнена аналитически или численными методами с помощью стандартных компьютерных программ. Тем не менее, хотелось бы иметь явное выражение, связывающее информационную емкость с параметрами канала. Для некоторых простых случаев это возможно, и рассмотрению этих случаев посвящен следующий параграф.

5.10. Симметричные каналы

Нашей задачей является дальнейшее упрощение формулы для информационной емкости канала. Сузив класс каналов до дискретных постоянных каналов (ДПК), задаваемых матрицей условных вероятностей $P = \{p(y \mid x), x \in X, y \in Y\}$, мы получили формулу

$$C_0 = \max_{\{p(x)\}} I(X;Y) . \qquad (5.10.1)$$

Рассмотрим некоторые частные случаи вида матрицы P.

ДПК называется *симметричным по входу*, если все строки его матрицы переходных вероятностей могут быть получены перестановками элементов первой строки.

ДПК называется *симметричным по выходу*, если все столбцы его матрицы переходных вероятностей могут быть получены перестановками элементов первого столбца.

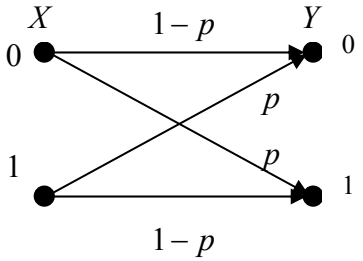
ДПК называется *полностью симметричным*, если он симметричен одновременно по входу и по выходу.

Пример 5.10.1. На рис. 5.10.1 показаны 3 диаграммы переходных вероятностей каналов и соответствующие матрицы переходных вероятностей. Три варианта моделей дают примеры полностью симметричного канала, симметричного только по входу и симметричного только по выходу. ■

Сформулируем те свойства симметричных каналов, которые упрощают вычисление их информационной емкости.

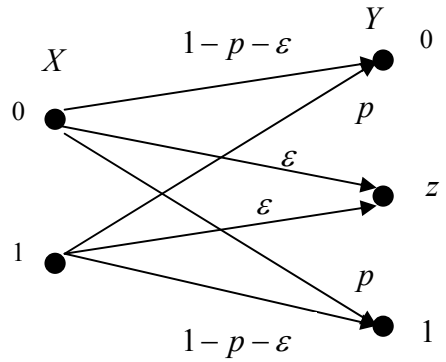
Свойство 5.10.1. Для симметричного по входу канала без памяти

$$C_0 = \max_{\{p(x)\}} \{H(Y)\} - H(Y \mid x), \; x \in X .$$



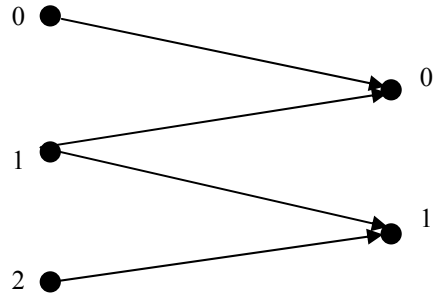
$$P = \begin{Vmatrix} 1-p & p \\ p & 1-p \end{Vmatrix}$$

а) ДСК



$$P = \begin{Vmatrix} 1-p-\varepsilon & \varepsilon & p \\ p & \varepsilon & 1-p-\varepsilon \end{Vmatrix}$$

б) ДСтК



$$P = \begin{Vmatrix} 1 & 0 \\ 1/2 & 1/2 \\ 0 & 1 \end{Vmatrix}$$

в) Симметричный по выходу канал

Рис.5.10.1. Примеры симметричных каналов

Свойство 5.10.2. Для симметричного по входу канала без памяти

$$C_0 \le \log L - H(Y \mid x),\ x \in X .$$

Свойство 5.10.3. Для симметричного по выходу канала без памяти при равновероятных входных символах выходные символы также равновероятны.

Свойство 5.10.4. Для полностью симметричного канала без памяти

$$C_0 = \log |Y| - H(Y \mid x),\ x \in X .$$

Доказательства. Начнем с первого свойства. Напомним, что

$$I(X,Y) = H(Y) - H(Y \mid X). \qquad (5.10.2)$$

В свою очередь вычитаемое можно записать как

$$H(Y \mid X) = \sum_x p(x) H(Y \mid x).$$

Поскольку все строки матрицы переходных вероятностей одинаковы с точностью до

нумерации элементов, все условные энтропии $H(Y \mid x)$ одинаковы. Следовательно, вычитаемое в (5.10.2) не зависит от входного распределения. Максимизация взаимной информации сводится к максимизации энтропии $H(Y)$, что и утверждается в свойстве 5.10.1.

Второе свойство следует из первого с учетом того, что энтропия ансамбля ансамбля не превышает логарифма числа его элементов (свойство 1…). Больше того, нам известно что равенство $H(Y) = \log|Y|$ имеет место при равновероятных буквах ансамбля Y. Свойство 5.10.3 доказывается с помощью формулы полной вероятности

$$p(y) = \sum_x p(x)p(y \mid x) .$$

При равновероятных буквах на входе

$$p(y) = \frac{1}{|X|}\sum_x p(y \mid x),\ y \in Y$$

Сумма в правой части представляет собой сумму элементов столбца матрицы переходных вероятностей. Поскольку в симметричном по выходу каналы столбцы одинаковы с точностью до перестановки их элементов, $p(y)$ не зависит от y и имеет место равенство $p(y) = 1/|Y|$. Из первых трех свойств немедленно следует свойство 5.10.4. ■

Пример 5.10.2. Из свойства 5.10.4 получаем формулу для информационной емкости двоичного симметричного канала (рис. 5.10.1а)

$$C_0 = 1 - h(p) .$$

График зависимости информационной емкости от переходной вероятности p показан на рис. 5.10.2. Информационная емкость равна нулю при $p = 1/2$. В этом случае вход и выход канала независимы. Естественно, с уменьшением p информационная емкость растет до значения $C_0 = 1$ при $p = 0$. Информационная емкость симметрична относительно точки $p = 1/2$. Это понятно, ведь при $p > 1/2$, переобозначив выходные символы, мы получим канал с переходной вероятностью $p < 1/2$. ■

К сожалению, сформулированных свойств недостаточно для того, чтобы плучить явные выражения для двоичного стирающего канала, показанного на рис. 5.10.1. Для этого симметричного по входу канала имеет место свойство 5.10.1, предполагающее оптимизацию по входному распределению. Однако, данная модель принадлежит классу симетричных каналов, для которых задача вычисления информационной емкости решается проще.

Канал называется *симметричным в широком смысле*, если перенумерацией выходных символов его матрица может быть представлена в форме клеточной матрицы

$$P = \left\| P_1 \middle| P_2 \middle| \ldots \middle| P_M \right\|, \qquad (5.10.3)$$

в которой каждая из подматриц P_i полностью симметрична (по входу и по выходу).

Пример 5.10.3. Матрица переходных вероятностей ДСтК (рис. 5.10.1) перенумерацией выходных символов (перестановкой столбцов) легко приводится к виду

$$P' = \left\| \begin{array}{cc|c} 1-p-\varepsilon & p & \varepsilon \\ p & 1-p-\varepsilon & \varepsilon \end{array} \right\| .$$

Обе подматрицы этой матрицы симметричны по входу и по выходу, следовательно, ДСтК симметричен в широком смысле. ■

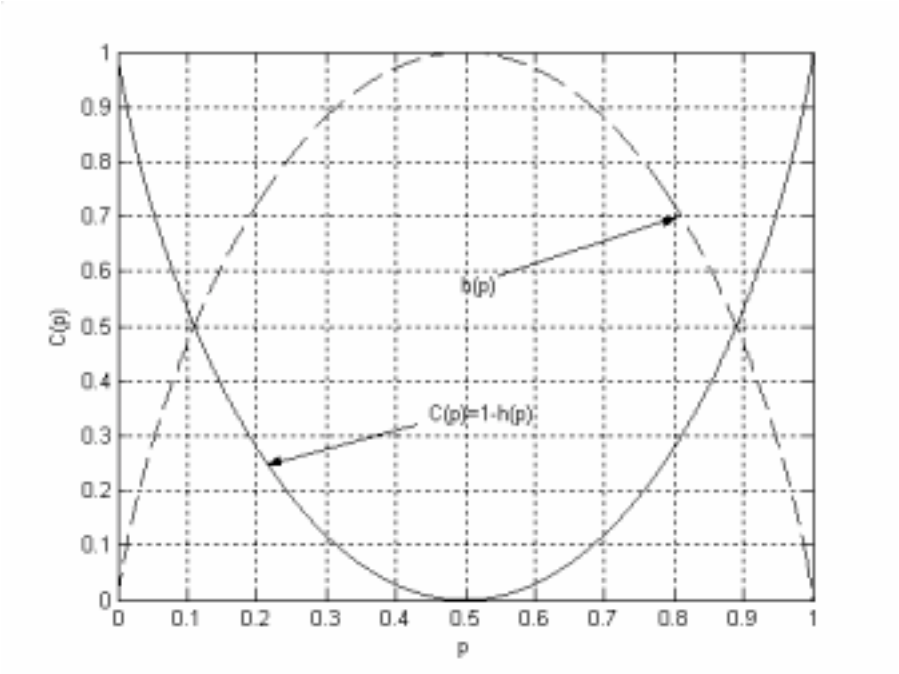


Рис. 5.10.2. Информационная емкость ДСК

Свойство 5.10.5. Для симметричного в широком смысле канала максимум взаимной информации между входом и выходом (см. (5.10.1)) достигается при равновероятных буквах входного алфавита.

Доказательство. Поскольку симметричный в широком смысле канал заведомо симметричен по входу, достаточно доказать, что равномерное распределение на входе канала максимизирует энтропию выходного алфавита Y.

Рассмотрим представление матрицы переходных вероятностей в форме (5.10.3) и обозначим через $Y_1,...,Y_M$ подмножества символов выходного алфавита Y, соответствующие подматрицам $P_1,...,P_M$. Энтропию ансамбля Y запишем в виде

$$H(Y) = -\sum_{i=1}^{M}\sum_{y\in Y_i} p(y)\log p(y). \qquad (5.10.3)$$

Введем обозначение

$$q_i = \sum_{y\in Y_i} p(y)$$

для вероятности подмножества Y_i. Набор чисел $q_1,...,q_M$ образует распределение вероятностей на множестве индексов $I = \{1,...,M\}$. Заметим, что для рассматриваемой модели канала вероятности $q_1,...,q_M$ не зависят от распределения $\{p(x)\}$ на множестве X. Чтобы убедиться в этом, запишем условную вероятность подмножества Y_i при известном символе x:

$$\mathrm{P}(Y_i \mid x) = \sum_{y\in Y_i} p(y \mid x)$$

Сумма в правой части по предположению не зависит от конкретного значения x, следовательно при всех x имеет место равенство $\mathrm{P}(Y_i \mid x) = q_i$.

Преобразуем (5.10.3) к виду

$$H(Y) = -\sum_{i=1}^{M} q_i \sum_{y\in Y_i} \frac{p(y)}{q_i}\log\left(\frac{p(y)}{q_i}q_i\right) = H(I) + \sum_{i=1}^{M} q_i H(Y_i), \qquad (5.10.4)$$

где

$$H(I) = -\sum_{i=1}^{M} q_i \log q_i \ -$$

энтропия множества индексов I, а

$$H(Y_i) = -\sum_{y \in Y_i} \frac{p(y)}{q_i} \log \frac{p(y)}{q_i} \ -$$

энтропия подмножества Y_i. Поскольку первое слагаемое в (5.10.4) не зависит от входного распределения, достаточно рассматривать второе слагаемое. Заметим, что, если все элементы подматрицы P_i поделить на вероятность q_i, то результатом будет стохастическая матрица, которая является матрицей переходных вероятностей полностью симметричного канала. Согласно свойству 5.10.3 максимальная энтропия выходного алфавита (равномерное распределение на выходе канала) достигается при равновероятных входных символах. Таким образом, при равновероятных символах на входе достигается максимум каждого слагаемого в правой части (5.10.4) и, следовательно, максимум энтропии $H(Y)$.■

Пример 5.10.4. Вернемся к модели ДСтК, представленной на рис.5.10.1б. Благодаря свойству 5.10.5 мы знаем, что информационная емкость этого канала равна взаимной информации между входом и выходом при равновероятных входных символах $p_x(0) = p_x(1) = 1/2$. Используя формулу полной вероятности, находим, что

$$p_y(0) = p_y(1) = \frac{1-\varepsilon}{2}, \ p_y(z) = \varepsilon .$$

Подставив эти значения в формулу для взаимной информации, после несложных преобразований можно получить формулу для информационной емкости

$$C_0 = (1-\varepsilon)\left(1 - h\left(\frac{p}{1-\varepsilon}\right)\right).$$

Из этой формулы, в частности следует очень простая формула для информационной емкости канала со стираниями и без ошибок. При $p = 0$ имеем

$$C_0 = 1 - \varepsilon .$$

Результат кажется очевидным. Поскольку доля «стертых» символов равна ε, то доля успешно переданных и при этом (в данном случае) заведомо правильно переданных символов как раз равна $1-\varepsilon$. Проблема, однако, состоит в том, что кодер не знает заранее, какие именно символы будут стерты. Тем не менее, полученная формула для информационной емкости говорит о том, что возможно такое кодирование, при котором скорость кода будет такой же, как если бы позиции стертых символов были известны заранее. При этом вероятность ошибки, как следует из доказываемой в следующем параграфе прямой теоремы кодирования, может быть сделана сколь угодно малой. ■

5.11. Прямая теорема кодирования для дискретных постоянных каналов

Нам предстоит доказать прямую теорему кодирования, утверждающую, что при скорости меньшей информационной емкости вероятность ошибки декодирования может быть сделана сколь угодно малой. Задача эта очень сложна. Прямой путь к ее решению состоит в построении последовательности кодов возрастающей длины n с убывающей по мере роста длины кодов вероятностью ошибки. Парадокс состоит в том, что таких конструкций кодов до сих пор не найдено, при том, что теорема кодирования, устанавливающая их существование, была успешно доказана Шенноном более полувека назад.

Метод, использованный Шенноном, называется методом случайного кодирования. Идея его очень проста. При заданной длине и скорости кода множество кодовых слов кода строится случайным выбором символов в соответствии с некоторым распределением

вероятностей. При таком построении кода сам код может рассматриваться как случайное событие, а его вероятность ошибки – как случайная величина. Предположим, что удалось найти математическое ожидание вероятности ошибки по множеству кодов и это значение оказалось равным некоторой величине ε. Тогда, очевидно, мы сможем утверждать, что, по меньшей мере для одного из кодов, вероятность ошибки не превышает ε.

Итак, воспользуемся методом случайного кодирования для доказательства прямой теоремы кодирования для дискретных постоянных каналов. Напомним, что рассматриваемый канал задается матрицей условных вероятностей $P = \{p(y \mid x), x \in X, y \in Y\}$ получения выходных символов $y \in Y$ при передаче по каналу символов $x \in X$. Поскольку ДПК – канал без памяти, для любой пары последовательностей $\mathbf{x} = (x_1,...,x_n) \in X^n$, $\mathbf{y} = (y_1,...,y_n) \in Y^n$ условная вероятность $p(\mathbf{y} \mid \mathbf{x})$ вычисляется как произведение побуквенных условных вероятностей:

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{n} p(y_i \mid x_i).$$

Код длины n объема M представляет собой произвольное подмножество $A = \{\mathbf{x}_m, m = 1,...,M\} \subseteq X^n$. Скорость кода равна

$$R = \frac{\log M}{n} \text{ бит/символ канала.}$$

Обозначим через $\hat{m}$ принятое декодером решение о номере переданного кодового слова при передаче кодового слова $\mathbf{x}_m$. Событие $\hat{m} \neq m$ представляет собой ошибку декодирования при передаче сообщения m, вероятность этого события обозначим как P_{em}. Считая все сообщения (номера кодовых слов) равновероятными, среднюю вероятность ошибки определим соотношением

$$P_e = \frac{1}{M} \sum_{m=1}^{M} P_{em}.$$

Теорема 5.11.1. Для дискретного постоянного канала с информационной емкостью C_0 для любых $\varepsilon, \delta > 0$ существует достаточно большое число n_0 такое, что для любого натурального числа $n \geq n_0$ существует код длины n со скоростью $R \geq C_0 - \delta$, средняя вероятность ошибки которого $P_e \leq \varepsilon$.

Иными словами, мы утверждаем, что при скорости кода сколь угодно близкой к информационной емкости C_0 (но, конечно, меньшей C_0 на сколь угодно малую величину δ) увеличением длины кодовых слов можно добиться сколь угодно малой вероятности ошибки (меньше любого ε).

Доказательство. Нам предстоит проделать следующий путь:

Шаг 1. Построить ансамбль случайных кодов с заданной длиной и скоростью.

Шаг 2. Указать правило декодирования.

Шаг 3. Оценить среднюю по ансамблю кодов вероятность ошибки и доказать, что вероятность ошибки убывает с увеличением длины кодов.

Построим ансамбль кодов. Обозначим через $\mathbf{p} = \{p(x), x \in X\}$ то распределение вероятностей на X, при котором имеет место равенство $C_0 = I(X,Y)$, т.е.

$$\mathbf{p} = \arg\max_{\{p(x)\}} I(X;Y).$$

Каждый код из ансамбля мы будем получать случайным и независимым выбором всех символов кодовых слов из множества X в соответствии с распределением вероятностей $\mathbf{p}$.

При заданной скорости кода R и его длине n число кодовых слов равно $M = 2^{nR}$. Исходя из условия теоремы, выбираем M таким, что

$$M - 1 < 2^{n(C_0 - \delta)} \le M \,. \qquad (5.11.1)$$

Перейдем к описанию работы декодера.

Формально правило принятия решений декодером задается разбиением всего множества Y^n на непересекающиеся решающие области R_m, $m = 1,...,M$, такие, что $\bigcup_{m=1}^{M} R_m = Y^n$, $R_m \bigcap R_{m'} = \varnothing$, $m' \neq m$. При получении на выходе канала последовательности $\mathbf{y} \in R_m$ декодер принимает решение в пользу сообщения с номером m (т.е. считает, что передавалось кодовое слово $\mathbf{x}_m$).

При равновероятных сообщениях оптимальным с точки зрения минимума средней вероятности ошибки является декодирование по принципу максимума правдоподобия (МП). При таком декодировании решение принимается в пользу такого кодового слова $\mathbf{x}_m$, для которого вероятность $p(\mathbf{y} \mid \mathbf{x}_m)$ максимальна. Вероятность $p(\mathbf{y} \mid \mathbf{x}_m)$ называют функцией правдоподобия кодового слова $\mathbf{x}_m$. Решающие области декодера по МП имеют вид

$$R_m = \{\mathbf{y} : p(\mathbf{y} \mid \mathbf{x}_m) \ge p(\mathbf{y} \mid \mathbf{x}_{m'}), m' \neq m\}.$$

Для доказательства прямой теоремы кодирования вместо оптимального декодирования по МП мы для простоты рассмотрим описанное ниже неоптимальное декодирование. На произведении множеств $X^n \times Y^n$ определим вспомогательное множество

$$T_n = \left\{(\mathbf{x}, \mathbf{y}) : \left| \frac{1}{n} I(\mathbf{x}; \mathbf{y}) - C_0 \right| \le \frac{\delta}{2} \right\}, \qquad (5.11.2)$$

где $I(\mathbf{x}; \mathbf{y})$ обозначает взаимную информацию между последовательностями $\mathbf{x}$ и $\mathbf{y}$.

По принятой из канала последовательности $\mathbf{y}$ декодер принимает решение в пользу $\mathbf{x}_m$ в том случае, если $(\mathbf{x}_m, \mathbf{y}) \in T_n$. Если таких кодовых слов несколько, выбирается любое из них. Если же такого слова нет, декодер выдает получателю произвольное решение.

Очевидно, описанный декодер существенно отличается от МП декодера и, следовательно, проигрывает ему по вероятности ошибки. В действительности, это станет яснее в процессе доказательства теоремы, рассматриваемый декодер принимает правильные решения в том случае, когда наблюдаемая последовательность $\mathbf{y}$ вместе с кодовым словом $\mathbf{x}_m$ образуют «типичную» пару. Типичность понимается здесь в точности в том же смысле, что и при обсуждении оптимального равномерного кодирования в части 1 курса.

Приступим к выводу оценки вероятности ошибки декодирования. Заметим, что ошибка возможна в двух случаях:

1. При передаче $\mathbf{x}_m$ наблюдается последовательность $\mathbf{y}$ такая, что $(\mathbf{x}_m, \mathbf{y}) \notin T_n$.
2. При передаче $\mathbf{x}_m$ на выходе канала имеем некоторую последовательность $\mathbf{y}$ такую, что для некоторого кодового слова $\mathbf{x}_{m'}$ $m' \neq m$ имеет место $(\mathbf{x}_{m'}, \mathbf{y}) \in T_n, m' \neq m$.

Вероятность первого из двух событий обозначим через P_{em1}, второго – через P_{em2}, соответствующие средние вероятности – через P_{e1} и P_{e2}. Для вероятности ошибки имеем оценку

$$P_e \le P_{e1} + P_{e2} \,. \qquad (5.11.3)$$

Знак неравенства использован потому, что два события, приводящие к ошибкам, не являются несовместными. Черту над вероятностями ошибок будем использовать для обозначения вероятностей, усредненных по множеству кодов. В частности, из (5.11.3) получаем

$$\overline{P}_e \le \overline{P}_{e1} + \overline{P}_{e2} \,. \qquad (5.11.4)$$

Оценим вероятность P_{e1}. Из определения множества T_n следует, что

$$P_{em1} = \mathrm{P}\left(\left| \frac{1}{n} I(\mathbf{x}_m; \mathbf{y}) - C_0 \right| > \frac{\delta}{2} \middle| \mathbf{x}_m \right), \tag{5.11.5}$$

где $\mathrm{P}(S \mid \mathbf{x_m})$ обозначает вероятность события S при передаче по каналу последовательности $\mathbf{x}_m$. Напомним, что P_{em1} – случайная величина, определенная на множестве кодов. Вместо оценки для величины P_{em1} мы будем вычислять оценку для ее среднего значения $\overline{P}_{em1}$.

Заметим, что, согласно (5.11.5), случайные значения величины P_{em1} определяются выбором только одного из кодовых слов случайного кода – слова $\mathbf{x}_m$, и не зависят от выбора других кодовых слов. Усредняя правую и левую части (5.11.5) по $\mathbf{x}_m$, получаем

$$\overline{P}_{em1} = \mathrm{P}\left(\left| \frac{1}{n} I(\mathbf{x}_m; \mathbf{y}) - C_0 \right| > \frac{\delta}{2} \right). \tag{5.11.6}$$

В этом равенстве в отличие от (5.11.5) последовательность $\mathbf{x}_m$ случайна, она получена случайным независимым выбором символов из ансамбля X в соответствии с распределением $\mathbf{p}$. Заметим также, что при независимых символах на входе канала без памяти символы на выходе канала также независимы (см. доказательство Теоремы 5.9.1). Следовательно,

$$I(\mathbf{x}_m; \mathbf{y}) = \sum_{i=1}^{n} I(x_{mi}; y_i) \text{ ,}$$

причем слагаемые в этой сумме независимы и одинаково распределены с математическим ожиданием

$$\mathrm{M}\left[I(x; y) \right] = C_0 .$$

В результате применения неравенства Чебышева для суммы независимых одинаково распределенных случайных величин приходим к оценке

$$\overline{P}_{em1} = \mathrm{P}\left(\left| \frac{1}{n} \sum_{i=1}^{n} I(x_{mi}; y) - C_0 \right| > \frac{\delta}{2} \right) \le \frac{4\mathbf{D}\left[I(x; y) \right]}{n\delta^2}. \tag{5.11.7}$$

Заметим, что выражение в правой части не зависит от m, поэтом оценка справедлива при всех m и также для средней по всем m вероятности $\overline{P}_{e1}$. Правая часть (5.11.7) убывает с ростом n и мы всегда можем выбрать значение длины кода $n = n_{01}$ так, чтобы при $n \ge n_{01}$ выполнялось неравенство

$$\overline{P}_{e1} \le \frac{\varepsilon}{2}, \quad n \ge n_{01} . \tag{5.11.8}$$

Теперь нужно оценить второе слагаемое в (5.11.4). Вероятность P_{em2} можно записать в виде

$$P_{em2} = \mathrm{P}\left(\text{существует } m' : (\mathbf{x}_{m'}; \mathbf{y}) \in T_n \middle| \mathbf{x}_m \right),$$

Аддитивной оценкой этой вероятности будет

$$P_{em2} \le \sum_{m' \ne m} \mathrm{P}\left((\mathbf{x}_{m'}; \mathbf{y}) \in T_n \middle| \mathbf{x}_m \right), \tag{5.11.9}$$

Найдем оценку среднего значения этой вероятности по множеству кодов. Введенный ансамбль кодов и модель канала задают совместное распределение вероятностей $\{ \tilde{p}(\mathbf{y}, \mathbf{x}_m, \mathbf{x}_{m'}), (\mathbf{y}, \mathbf{x}_m, \mathbf{x}_{m'}) \in Y^n \times X^n \times X^n \}$. В этом ансамбле кодовые слова $\mathbf{x}_m$ и $\mathbf{x}_{m'}$ независимы, а последовательность на выходе канала $\mathbf{y}$ зависит только от переданного кодового слова $\mathbf{x}_m$. Поэтому

$$\tilde{p}(\mathbf{y}, \mathbf{x}_m, \mathbf{x}_{m'}) = p(\mathbf{y}, \mathbf{x}_m) p(\mathbf{x}_{m'}) .$$

Оценка вероятности P_{em2} (5.11.9) эквивалентна неравенству

$$\overline{P}_{em2} \le \sum_{m' \ne m} \sum_{\mathbf{x}_{m'}} \sum_{\mathbf{x}_m} \sum_{\mathbf{y}:(\mathbf{x}_{m'},\mathbf{y}) \in T_n} \widetilde{p}(\mathbf{y}, \mathbf{x}_m, \mathbf{x}_{m'}) .$$

Меняя порядок суммирования и учитывая, что

$$\sum_{\mathbf{x}_m} \widetilde{p}(\mathbf{y}, \mathbf{x}_m, \mathbf{x}_{m'}) = \sum_{\mathbf{x}_m} p(\mathbf{y}, \mathbf{x}_m) p(\mathbf{x}_{m'}) = p(\mathbf{y}) p(\mathbf{x}_{m'}) ,$$

приходим к более простой оценке

$$\overline{P}_{em2} \le \sum_{m' \ne m} \sum_{\mathbf{x}_{m'}} \sum_{\mathbf{y}:(\mathbf{x}_{m'},\mathbf{y}) \in T_n} p(\mathbf{y}) p(\mathbf{x}_{m'}) . \qquad (5.11.10)$$

Для пар $(\mathbf{x}_{m'}, \mathbf{y}) \in T_n$ согласно (5.11.2) имеет место неравенство

$$I(\mathbf{x}_{m'}, \mathbf{y}) = \log \frac{p(\mathbf{x}_{m'}, \mathbf{y})}{p(\mathbf{x}_{m'}) p(\mathbf{y})} \ge n(C_0 - \tfrac{\delta}{2}) ,$$

из которого получаем

$$p(\mathbf{y}) p(\mathbf{x}_{m'}) \le p(\mathbf{x}_{m'}, \mathbf{y}) 2^{-n(C_0 - \delta/2)} .$$

Подставив этот результат в (5.11.10) и распространив суммирование на все последовательности $\mathbf{y}$, получим

$$\overline{P}_{em2} \le \sum_{m' \ne m} \sum_{\mathbf{x}_{m'}} \sum_{\mathbf{y}} p(\mathbf{x}_{m'}, \mathbf{y}) 2^{-n(C_0 - \delta/2)} .$$

Из условия нормировки вероятностей получаем

$$\overline{P}_{em2} \le \sum_{m' \ne m} 2^{n(C_0 + \delta/2)} = (M-1) 2^{n(C_0 + \delta/2)} .$$

Воспользовавшись (5.11.1), приходим к неравенству

$$\overline{P}_{em2} \le 2^{n(C_0 - \delta)} 2^{-n(C_0 - \delta/2)} = 2^{-n\delta/2} .$$

Правая часть убывает с увеличением длины кода n, и найдется длина n_{02} такая, что

$$\overline{P}_{em2} \le \frac{\varepsilon}{2}, \quad n > n_{02} . \qquad (5.11.11)$$

Заметим, что индекс m можно опустить, поскольку правая часть него не зависит. С учетом (5.11.11) и (5.11.8) из (5.11.4) следует, что при $n \ge \max\{n_{01}, n_{02}\}$ средняя по ансамблю кодов вероятность ошибки не превышает ε. Как уже говорилось, отсюда вытекает, что хотя бы один из кодов ансамбля имеет вероятность ошибки не больше ε. Тем самым теорема доказана. ■