БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра РЭС

Реферат на тему:

**«Коды без памяти. Коды Хаффмена. Коды с памятью»**

МИНСК, 2009

**Коды без памяти. Коды Хаффмена**

Простейшими кодами, на основе которых может выполняться сжатие данных, являются коды без памяти. В коде без памяти *каждый символ в кодируемом векторе данных заменяется кодовым словом из префиксного множества двоичных последовательностей или слов.*

*Префиксным множеством двоичных последовательностей* ***S*** *называется конечное множество двоичных последовательностей, таких, что ни одна последовательность в этом множестве не является префиксом, или началом, никакой другой последовательности в* ***S****.*

К примеру, множество двоичных слов ***S1 = {00, 01, 100, 110, 1010, 1011}*** является префиксным множеством двоичных последовательностей, поскольку, если проверить любую из 30 возможных совместных комбинаций (***wi wj***) из ***S1,*** то видно, что ***wi*** никогда не явится префиксом (или началом) ***wj***. С другой стороны, множество ***S2 = { 00, 001, 1110 }*** не является префиксным множеством двоичных последовательностей, так как последовательность 00 является префиксом (началом) другой последовательности из этого множества - 001.

Таким образом, если необходимо закодировать некоторый вектор данных ***X***= ***( x1, x2,… xn )*** с алфавитом данных Aразмера ***k***, то *кодирование кодом без памяти осуществляется следующим образом:*

- составляют полный список символов ***a1, a2, aj ... ,ak*** алфавита A, в котором ***aj*** - ***j***-й по частоте появления в ***X*** символ, то есть первым в списке будет стоять наиболее часто встречающийся в алфавите символ, вторым – реже встречающийся и т.д.**;**

- каждому символу ***aj*** назначают кодовое слово ***wj*** из префиксного множества двоичных последовательностей ***S***;

- выход кодера получают объединением в одну последовательность всех полученных двоичных слов.

Формирование префиксных множеств и работа с ними – это отдельная серьезная тема из теории множеств, выходящая за рамки нашего курса, но несколько необходимых замечаний все-таки придется сделать.

Если ***S*** = { ***w1, w2, ... , wk*** } - префиксное множество, то можно определить некоторый вектор ***v(S)*** = ( ***L1, L2, ... , Lk*** ), состоящий из чисел, являющихся длинами соответствующих префиксных последовательностей (***Li*** - длина ***wi*** ).

Вектор (***L1, L2, ... , Lk***), состоящий из неуменьшающихся положительных целых чисел, называется вектором Крафта. Для него выполняется неравенство

. (1)

Это неравенство называется неравенством Крафта. Для него справедливо следующее утверждение: если ***S*** - какое-либо префиксное множество, то ***v(S)*** - вектор Крафта.

Иными словами, *длины двоичных последовательностей в префиксном множестве удовлетворяют неравенству Крафта.*

Если неравенство (1) переходит в строгое равенство, то такой код называется компактным и обладает наименьшей среди кодов с данным алфавитом длиной, то есть является оптимальным.

Ниже приведены примеры простейших префиксных множеств и соответствующие им векторы Крафта:

***S1 = {0, 10, 11} и v(S1) = ( 1, 2, 2 );***

***S2 = {0, 10, 110, 111}*** и ***v(S2) = ( 1, 2, 3, 3 );***

***S3 = {0, 10, 110, 1110, 1111}*** и ***v(S3) = ( 1, 2, 3, 4, 4 );***

***S4 = {0, 10, 1100, 1101, 1110, 1111}*** и ***v(S4) = ( 1, 2, 4, 4, 4, 4 );***

***S5 = {0, 10, 1100, 1101, 1110, 11110, 11111}*** и ***v(S5) = ( 1, 2, 4, 4, 4, 5, 5 );***

***S6 = {0, 10, 1100, 1101, 1110, 11110, 111110, 111111}***

и ***v(S6) = (1,2,4,4,4,5,6,6).***

Допустим мы хотим разработать код без памяти для сжатия вектора данных ***X = ( x1, x2,… xn )*** с алфавитом Aразмером в ***k*** символов. Введем в рассмотрение так называемый вектор частот ***F = ( F1, F2, ... , Fk ),*** где ***Fi*** - количество появлений ***i***-го наиболее часто встречающегося символа из A в ***X***. Закодируем ***X*** кодом без памяти, для которого вектор Крафта ***L = ( L1, L2, ... , Lk )***.

Тогда длина двоичной кодовой последовательности ***B(X)*** на выходе кодера составит

***L\*F = L1\*F1 + L2\*F2 + ... + Lk\*Fk .***(2)

Лучшим кодом без памяти был бы код, для которого длина ***B(X)*** - минимальна. Для разработки такого кода нам нужно найти вектор Крафта ***L,*** для которого произведение ***L\*F*** было бы минимальным.

Простой перебор возможных вариантов - вообще-то, не самый лучший способ найти такой вектор Крафта, особенно для большого **k**.

Алгоритм Хаффмена, названный в честь его изобретателя - Дэвида Хаффмена, - дает нам эффективный способ поиска оптимального вектора Крафта **L** для **F**, то есть такого **L**, для которого точечное произведение **L\*F** – минимально. Код, полученный с использованием оптимального **L** для **F**, называют кодом Хаффмена.

### **Алгоритм Хаффмена**

Алгоритм Хаффмена изящно реализует общую идею статистического кодирования с использованием префиксных множеств и работает следующим образом:

1. Выписываем в ряд все символы алфавита в порядке возрастания или убывания вероятности их появления в тексте.

2. Последовательно объединяем два символа с наименьшими вероятностями появления в новый составной символ, вероятность появления которого полагаем равной сумме вероятностей составляющих его символов. В конце концов построим дерево, каждый узел которого имеет суммарную вероятность всех узлов, находящихся ниже него.

3. Прослеживаем путь к каждому листу дерева, помечая направление к каждому узлу (например, направо - 1, налево - 0) . Полученная последовательность дает кодовое слово, соответствующее каждому символу (рис. 1).

Построим кодовое дерево для сообщения со следующим алфавитом:

 **A B C D E**

 10 5 8 13 10

 **B C A E D**

 5 8 10 10 13

 **A E BC D**

 10 10 13 13

 **BC D AE**

 13 13 20

 **AE BCD**

 20 26

 **AEBCD**

0

1

C

B

0

0

0

1

1

1

E

A

D

 46

Рис. 1

*Недостатки метода Хаффмена*

Самой большой сложностью с кодами Хаффмена, как следует из предыдущего обсуждения, является необходимость иметь таблицы вероятностей для каждого типа сжимаемых данных. Это не представляет проблемы, если известно, что сжимается английский или русский текст; мы просто предоставляем кодеру и декодеру подходящее для английского или русского текста кодовое дерево. В общем же случае, когда вероятность символов для входных данных неизвестна, статические коды Хаффмена работают неэффективно.

Решением этой проблемы является статистический анализ кодируемых данных, выполняемый в ходе первого прохода по данным, и составление на его основе кодового дерева. Собственно кодирование при этом выполняется вторым проходом.

Существует, правда, динамическая версия сжатия Хаффмена, которая может строить дерево Хаффмена "на лету" во время чтения и активного сжатия. Дерево постоянно обновляется, чтобы отражать изменения вероятностей входных данных. Однако и она на практике обладает серьезными ограничениями и недостатками и, кроме того, обеспечивает меньшую эффективность сжатия.

Еще один недостаток кодов Хаффмена - это то, что минимальная длина кодового слова для них не может быть меньше единицы, тогда как энтропия сообщения вполне может составлять и 0,1, и 0,01 бит/букву. В этом случае код Хаффмена становится существенно избыточным. Проблема решается применением алгоритма к блокам символов, но тогда усложняется процедура кодирования/декодирования и значительно расширяется кодовое дерево, которое нужно в конечном итоге сохранять вместе с кодом.

Наконец, код Хаффмена обеспечивает среднюю длину кода, совпадающую с энтропией, только в том случае, когда вероятности символов источника являются целыми отрицательными степенями двойки: 1/2 = 0,5; 1/4 = 0,25; 1/8 = 0,125; 1/16 = 0,0625 и т.д. На практике же такая ситуация встречается очень редко или может быть создана блокированием символов со всеми вытекающими отсюда последствиями.

**Коды с памятью**

Обычно рассматривают два типа кодов с памятью:

*- блочные коды;*

*- коды с конечной памятью.*

*Блочный код* делит вектор данных на блоки заданной длины и затем каждый блок заменяют кодовым словом из префиксного множества двоичных слов. Полученную последовательность кодовых слов объединяют в результирующую двоичную строку на выходе кодера. О блочном коде говорят, что он - блочный код ***k***-го порядка, если все блоки имеют длину, равную ***k***.

В качестве примера сожмем вектор данных ***X*** = (0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1), используя блочный код второго порядка.

Сначала разобьем вектор ***X*** на блоки длины 2: 01, 10, 10, 01, 11, 10, 01, 01.

Будем рассматривать эти блоки как элементы нового “гипералфавита” {01, 10, 11}. Чтобы определить, какой код назначить какому их символов этого нового алфавита, определим вектор частот появлений элементов дополнительного алфавита в последовательности блоков. Получим вектор частот (4, 3, 1), то есть наиболее часто встречающийся блок 01 появляется 4 раза, следующий по частоте встречаемости блок 10 - три раза и наименее часто встречающийся блок 11 - лишь один раз.

Оптимальный вектор Крафта для вектора частот ( 4, 3, 1 ) - это вектор (1, 2, 2). Таким образом, кодер для оптимального блочного кода второго порядка применительно к заданному вектору данных ***X*** определяется табл. 1.

Таблица 1

|  |
| --- |
| Таблица кодера |
| Блок | Кодовое слово |
| 01 | 0 |
| 10 | 10 |
| 11 | 11 |

Заменяя каждый блок присвоенным ему кодовым словом из таблицы кодера, получим последовательность кодовых слов:

0, 10, 10, 0, 11, 10, 0, 0.

Объединяя эти кодовые слова вместе, имеем выходную последовательность кодера:

***B(X)*** = ( 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0 ).

Можно проверить, что энтропия ***H(X)*** исходного вектора ***X*** равна ***0.9887*** бит/букву, а степень сжатия, получаемая в результате использования блочного кода ***R(X)*** = 12/16 = 0.75 бит на выборку данных, оказалась меньше нижней границы энтропии. Полученный результат, казалось бы, противоречит теореме Шеннона, утверждающей, что невозможно достичь средней длины кода, меньшей энтропии источника. Однако на самом деле это не так. Если внимательно посмотреть на вектор данных ***X***, то можно заметить закономерность: за символом 0 чаще следует 1, то есть условная вероятность ***Р(1/0)*** существенно больше, чем просто ***Р(0).*** Следовательно, энтропию этого источника нужно считать как энтропию сложного сообщения, а она, как известно, всегда меньше, чем для источника простых сообщений.

*Код с конечной памятью* при кодировании вектора данных ***( X1, X2, ..., Xn*** ) использует кодовую книгу, состоящую *из нескольких различных кодов без памяти*. Каждая выборка данных кодируется кодом без памяти из кодовой книги, определяемым значением некоторого числа предыдущих выборок данных.

В качестве примера кодирования кодом с памятью рассмотрим уже упоминавшийся ранее классический пример ***X = ABRACADABRA*.** В этой последовательности совершенно очевидна сильная статистическая зависимость между ее очередными символами, которая должна обязательно учитываться при выборе метода кодирования.

Кодер с памятью *при кодировании текущего символа учитывает значение предшествующего ему символа*. Таким образом, кодовое слово для текущего символа ***A*** будет различным в сочетаниях ***RA, DA*** и ***CA*** (иными словами, код обладает памятью в один символ источника) (табл. 2).

Таблица 2

|  |
| --- |
| ***Кодер*** |
| Символ, предыдущий символ | Кодовое слово |
| (A,-) | 1 |
| (B,A) | 0 |
| (C,A) | 10 |
| (D,A) | 11 |
| (A,R) | 1 |
| (R,B) | 1 |
| (A,C) | 1 |
| (A,B) | 1 |

Результат кодирования - вектор ***B(X) = (10111011111011)*** длиной в 11 бит искоростью сжатия ***R*** ***= 13/11=1,18*** бита на символ данных, тогда как при кодировании равномерным трехразрядным кодом с ***R = 3*** понадобилось бы 33 бита.

**ЛИТЕРАТУРА**

1. Лидовский В.И. Теория информации. - М., «Высшая школа», 2002г. – 120с.
2. Метрология и радиоизмерения в телекоммуникационных системах. Учебник для ВУЗов. / В.И.Нефедов, В.И.Халкин, Е.В.Федоров и др. – М.: Высшая школа, 2001 г. – 383с.
3. Цапенко М.П. Измерительные информационные системы. – М.: Энергоатом издат, 2005. - 440с.
4. Зюко А.Г. , Кловский Д.Д., Назаров М.В., Финк Л.М. Теория передачи сигналов. М: Радио и связь, 2001 г. –368 с.
5. Б. Скляр. Цифровая связь. Теоретические основы и практическое применение. Изд. 2-е, испр.: Пер. с англ. – М.: Издательский дом «Вильямс», 2003 г. – 1104 с.