Кодирование

В.Е. Алексеев

1. Задача оптимального кодирования

Побуквенное кодирование. Пусть $A=\{a_1,a_2,\ldots,a_k\}$ и $B=\{b_1,b_2,\ldots,b_q\}$ – два алфавита. *Побуквенное кодирование* состоит в том, что в кодируемом тексте (слове в алфавите A) каждая буква алфавита A заменяется словом в алфавите B. Эти замены производятся в соответствии с *кодом*, указывающим для каждой буквы слово, которым она заменяется:

$a_1 \to u_1,$
$a_2 \to u_2,$
$\ldots$
$a_k \to u_k.$

Здесь $u_1,u_2,\ldots,u_k$ – слова в алфавите B. Таким образом, слово $\alpha = a_{i_1}a_{i_2}\ldots a_{i_n}$ кодируется словом $f(\alpha)=u_{i_1}u_{i_2}\ldots u_{i_n}$. Кодируемое слово α будем называть *сообщением*, а результат кодирования $f(\alpha)$ – *кодограммой*.

Считая алфавит A линейно упорядоченным, мы можем задать код как упорядоченный набор слов в алфавите B: $U=(u_1,u_2,\ldots,u_k)$. Его элементы называются *кодовыми словами*.

Обратимый код. Для того, чтобы любую кодограмму можно было однозначно декодировать, функция f должна быть инъективной, т.е. для любых двух различных слов α и β в алфавите A слова $f(\alpha)$ и $f(\beta)$ тоже должны быть различными. Код, для которого это выполняется, называется *обратимым*.

Если f не инъективна, то найдутся такие слова $\alpha = a_{i_1}a_{i_2}\ldots a_{i_n}$ и $\beta = a_{j_1}a_{j_2}\ldots a_{j_m}$, что $(i_1,i_2,\ldots,i_n)\neq(j_1,j_2,\ldots j_m)$ (т.е. эти слова различны) и $f(\alpha)=f(\beta)$, т.е. $u_{i_1}u_{i_2}\ldots u_{i_n}=u_{j_1}u_{j_2}\ldots u_{j_m}$. Поэтому определение обратимости равносильно следующему: код $U=(u_1,u_2,\ldots,u_k)$ обратим, если равенство $u_{i_1}u_{i_2}\ldots u_{i_n}=u_{j_1}u_{j_2}\ldots u_{j_m}$ имеет место тогда и только тогда, когда $(i_1,i_2,\ldots,i_n)=(j_1,j_2,\ldots j_m)$.

Коэффициент сжатия. Допустим, известно распределение вероятностей $P=(p_1,p_2,\ldots,p_k)$ появления букв $a_1,a_2,\ldots,a_k$ в кодируемых текстах. Считаем, что $p_i>0$ для всех $i=1,\ldots,k$.

Если код $U=(u_1,u_2,\ldots,u_k)$ применяется для кодирования сообщения α длины n, в которое буква a_1 входит n_1 раз, буква a_2 – n_2 раз, …, буква a_k – n_k раз, то длина кодограммы будет $n_1|u_1|+n_2|u_2|+\ldots+n_k|u_k|$ (прямые скобки обозначают длину слова). При большом n частоты вхождения букв $\frac{n_i}{n}$ будут мало отличаться от соответствующих вероятностей и отношение длины кодограммы к длине сообщения будет близко к

величине $\sum_{i=1}^{k} p_i |u_i|$. Эта величина называется коэффициентом сжатия кода U при данном распределении вероятностей P и обозначается через $C(P,U)$.

Постановка задачи. Даны алфавиты $A=\{a_1,a_2,\ldots,a_k\}$ и $B=\{b_1,b_2,\ldots,b_q\}$ и распределение вероятностей $P=(p_1,p_2,\ldots,p_k)$ букв алфавита A. Требуется найти обратимый код $U=(u_1,u_2,\ldots,u_k)$ с наименьшим коэффициентом сжатия $C(P,U)$.

2. Префиксные коды

Код называется *префиксным,* если ни одно из кодовых слов не является началом (*префиксом*) другого кодового слова.

Лемма 1. *Всякий префиксный код является обратимым.*

Доказательство. Если код $U=(u_1,u_2,\ldots,u_k)$ не обратим, то существуют такие наборы $(i_1,i_2,\ldots,i_n)$ и $(j_1,j_2,\ldots j_m)$, что $(i_1,i_2,\ldots,i_n)\neq(j_1,j_2,\ldots j_m)$ и $u_{i_1}u_{i_2}\ldots u_{i_n}=u_{j_1}u_{j_2}\ldots u_{j_m}$. Найдем наименьшее s, при котором $i_s\neq j_s$. Тогда одно из слов u_{i_s} и u_{j_s} является началом другого, следовательно, код U не префиксный.

Следующие две теоремы позволяют свести задачу построения оптимального обратимого кода к задаче построения оптимального префиксного кода.

Теорема 1. (неравенство Макмиллана). *Если $U=(u_1,u_2,\ldots,u_k)$ – обратимый код в алфавите из q букв, то выполняется неравенство* $\sum_{i=1}^{k} q^{-|u_i|}\le 1$.

Доказательство. Рассмотрим величину $S_n=\sum_{|\alpha|=n} q^{-|f(\alpha)|}$. Очевидно, $S_1=\sum_{i=1}^{k} q^{-|u_i|}$ и мы должны доказать, что $S_1\le 1$. При произвольном n имеем

$$S_n=\sum_{a_{i_1}a_{i_2}\ldots a_{i_n}} q^{-|u_{i_1}u_{i_2}\ldots u_{i_n}|}=\sum_{i_1=1}^{k}\sum_{i_2=1}^{k}\ldots\sum_{i_{n=1}}^{k} q^{-|u_{i_1}|}q^{-|u_{i_2}|}\ldots q^{-|u_{i_n}|}=\left(\sum_{i_1=1}^{k} q^{-|u_{i_1}|}\right)\left(\sum_{i_2=1}^{k} q^{-|u_{i_2}|}\right)\ldots\left(\sum_{i_1=1}^{k} q^{-|u_{i_n}|}\right)=S_1^n. \quad (1)$$

Обозначим через $m(n,l)$ число сообщений длины n, имеющих кодограмму длины l. Группируя в сумме S_n слагаемые с одинаковым показателем степени, получаем

$$S_n=\sum_{i=1}^{nL} m(n,l)q^{-l}.$$

Так как код обратимый, то кодограммы различных сообщений различны, поэтому $m(n,l)\le q^l$. Следовательно, $S_n\le nL$. Учитывая (1), видим, что при любом n выполняется неравенство

$$S_1 \le (nL)^{\frac{1}{n}}.$$

Оно будет верно и в пределе при $n \to \infty$, а предел правой части равен 1. Теорема доказана.

Теорема 2. *Пусть* $l_1, l_2, \ldots, l_k$ *– натуральные числа, удовлетворяющие неравенству* $\sum_{i=1}^{k} q^{-l_i} \le 1$. *Тогда существует префиксный код* $U = (u_1, u_2, \ldots, u_k)$ *в алфавите из* q *букв, в котором* $|u_i| = l_i$, $i = 1, \ldots, k$.

Доказательство. Не теряя общность, можно предположить, что $l_1 \le l_2 \le \ldots \le l_k$. Будем строить код U, последовательно добавляя слова длины $l_1, l_2, \ldots, l_k$ и следя за тем, чтобы после добавления очередного слова код оставался префиксным.

В качестве u_1 берем любое слово длины l_1 в алфавите $B = \{b_1, b_2, \ldots, b_q\}$ и образуем код $U_1 = \{u_1\}$. Он, очевидно, префиксный. Допустим, уже построен префиксный код $U_{i-1} = \{u_1, \ldots, u_{i-1}\}$ с длинами слов $l_1, \ldots, l_{i-1}$. Составим список всех слов длины l_i в алфавите B, этот список состоит из q^{l_i} слов. Вычеркнем из списка все слова, у которых префикс принадлежит множеству U_{i-1}. Так как U_{i-1} – префиксный код, то каждое слово будет вычеркнуто не более одного раза. Число слов с префиксом u_j в этом списке равно $q^{l_i - l_j}$, значит, всего будет вычеркнуто $q^{l_i - l_1} + \ldots + q^{l_i - l_{i-1}}$ слов. Покажем, что после этого список будет непустым, т.е. $q^{l_i} - q^{l_i - l_1} - \ldots - q^{l_i - l_{i-1}} \ge 1$. Действительно, это неравенство равносильно $q^{-l_1} + \ldots + q^{-l_{i-1}} + q^{-l_i} \le 1$, а последнее следует из условия теоремы. Таким образом, список непустой и мы можем взять из него любое слово и добавить его к коду U_{i-1}. Новый код U_i тоже будет префиксным.

Следствие. Для любого распределения вероятностей $P = (p_1, p_2, \ldots, p_k)$ и любого обратимого кода $U = (u_1, u_2, \ldots, u_k)$ существует такой префиксный код V в том же алфавите, что $C(P,V) = C(P,U)$.

Доказательство. Если код U обратимый, то для него выполняется неравенство Макмиллана $\sum_{i=1}^{k} q^{-|u_i|} \le 1$. По теореме 2 существует префиксный код $V = (v_1, v_2, \ldots, v_k)$ такой, что $|v_i| = |u_i|$ для $i = 1, \ldots, k$. Но тогда $\sum_{i=1}^{k} p_i |v_i| = \sum_{i=1}^{k} p_i |u_i|$, т.е. $C(P,V) = C(P,U)$.

Таким образом, при любом распределении вероятностей P среди обратимых кодов с наименьшим коэффициентом сжатия имеется хотя бы один префиксный код. Поэтому в при решении задачи построения оптимального кода можно ограничиться рассмотрением префиксных кодов. Далее излагается метод решения задачи построения оптимального префиксного кода.

3. Метод Хаффмена

Далее предполагается, что буквы алфавита A упорядочены по невозрастанию вероятностей: $p_1 \geq p_2 \geq \ldots \geq p_k$.

Лемма 2. *Если* $p_1 \geq p_2 \geq \ldots \geq p_k$, *то существует оптимальный префиксный код* $U = (u_1, u_2, \ldots, u_k)$ *такой, что* $|u_1| \leq |u_2| \leq \ldots \leq |u_k|$.

Доказательство. Пусть $U = (u_1, u_2, \ldots, u_k)$ оптимальный префиксный код и допустим, что при некоторых i и j, $i < j$, имеет место $|u_i| > |u_j|$. Рассмотрим код U', получающийся из кода U перестановкой слов u_i и u_j (напомним, что код – это упорядоченный набор слов). Тогда

$$C(P,U) - C(P,U') = p_i |u_i| + p_j |u_j| - p_i |u_j| - p_j |u_i| = (p_i - p_j)(|u_i| - |u_j|) \geq 0,$$

т.е. $C(P,U') \leq C(P,U)$. Так как код U оптимальный, а код U' – префиксный, то может быть только равенство: $C(P,U') = C(P,U)$. Значит, U' – тоже оптимальный код. Повторяя такие перестановки слов, в конце концов получим оптимальный код, в котором слова располагаются в порядке неубывания длин.

Приступая к решению задачи построения оптимального префиксного кода, начнем со случая, когда алфавит B состоит из двух букв: $B = \{0,1\}$, $q = 2$ (двоичный код). Метод решения основан на сведении задачи для данного алфавита A к той же задаче для алфавита с меньшим числом букв. Это сведение описывает и обосновывает следующая теорема, называемая теоремой редукции.

Пусть задано распределение вероятностей $P = (p_1, p_2, \ldots, p_k)$ букв алфавита $A = \{a_1, a_2, \ldots, a_k\}$. Определим *редуцированный алфавит* $A' = \{a_1, a_2, \ldots a_{k-2}, b\}$ с редуцированным распределением $P' = (p_1, p_2, \ldots, p_{k-2}, p_{k-1} + p_k)$. Это можно трактовать как замену каждой из букв a_{k-1} и a_k новой буквой b.

Теорема редукции. *Если* $U' = (u_1, \ldots, u_{k-2}, z)$ *– оптимальный префиксный код для алфавита* A' *с распределением* P', *то* $U = (u_1, \ldots, u_{k-2}, z0, z1)$ *– оптимальный префиксный код для алфавита* A *с распределением* P.

Доказательство. Допустим, что это не так и оптимальным префиксным кодом для распределения P является код $V = (v_1, \ldots, v_k)$, тогда $C(P,V) < C(P,U)$. Ввиду леммы 2 можно считать, что $|v_1| \leq \ldots \leq |v_k|$. Допустим для определенности, что последнее слово в коде V оканчивается буквой 0: $v_k = w0$. Рассмотрим код $W = (v_1, \ldots, v_{k-1}, w)$. Так как, очевидно, $C(P,W) < C(P,V)$, то код W не может быть префиксным. Единственной причиной этого может быть то, что в этом коде имеется слово v_i с префиксом w. Тогда $v_i = w1$. Все слова от v_i до v_k имеют одинаковую длину. Не теряя общности, можно считать, что $i = k-1$ (иначе переставим слова v_i и v_{k-1} и получим код с тем же коэффициентом сжатия). Таким образом, $V = (v_1, \ldots, v_{k-2}, w1, w0)$. Рассмотрим код $V' = (v_1, \ldots, v_{k-2}, w)$ и вычислим разность

$$C(P,V)-C(P',V')=p_{k-1}(|w|+1)+p_k(|w|+1)-(p_{k-1}+p_k)|w|=p_{k-1}+p_k.$$

Такова же величина разности $C(P,U)-C(P',U')$. Следовательно,

$$C(P,V)-C(P,U)=C(P',V')-C(P',U')$$

и если код U не является оптимальным кодом для распределения P, то и код U' не является оптимальным кодом для распределения P'. Теорема доказана.

Алгоритм построения оптимального префиксного кода, основанный на этой теореме, удобно представить в графической форме. Рассмотрим бесконечное корневое бинарное дерево. В этом дереве из каждой вершины в следующий ярус выходят два ребра, одно из которых считается левым, другое – правым. Каждому левому ребру приписывается буква 0, каждому правому – буква 1. Двигаясь вдоль какого-нибудь пути в этом дереве, можно прочитать некоторое слово в алфавите {0,1}. Каждой вершине дерева поставим в соответствие слово, читающееся вдоль пути из корня в эту вершину (самому корню соответствует пустое слово). Получается взаимно однозначное соответствие между вершинами дерева и словами в алфавите {0,1}.

Пусть $U=(u_1,u_2,\ldots,u_k)$ – префиксный код. Рассмотрим фрагмент бесконечного бинарного дерева, состоящий из вершин, соответствующих словам кода и всех вершин, лежащих на путях, соединяющих их с корнем. Это конечное дерево является графическим представлением кода U. Обратно, каждое конечное бинарное дерево представляет некоторый префиксный код. Слова кода представляются листьями дерева, а длина слова есть длина пути из соответствующей вершины к корню. Задача построения оптимального кода теперь может быть сформулирована как задача построения для данного набора чисел $P=(p_1,p_2,\ldots,p_k)$ бинарного дерева с k листьями $a_1,a_2,\ldots,a_k$ и минимальным значением величины $\sum_{i=1}^{k}p_il_i$, где l_i – расстояние от вершины a_i до корня. Теорема редукции обосновывает следующий алгоритм решения этой задачи.

Алгоритм Хаффмена.

Дано: положительные числа $p_1,p_2,\ldots,p_k$

Результат: бинарное дерево с k листьями $a_1,a_2,\ldots,a_k$; каждой вершине x этого дерева приписано число $p(x)$, причем $p(a_i)=p_i$.

1. Создать вершины $a_1,a_2,\ldots,a_k$;
   положить $p(a_i)=p_i$, $i=1,\ldots,k$;
   положить $A=\{a_1,a_2,\ldots,a_k\}$.
2. Пока $A\neq\varnothing$, повторять
   2.1. Создать новую вершину z.
   2.2. Выбрать в A вершину x с наименьшим значением $p(x)$;
   сделать x левым сыном вершины z;
   удалить x из A.
   2.3. Выбрать в A вершину y с наименьшим значением $p(y)$;
   сделать y правым сыном вершины z;
   удалить y из A.
   2.4. Положить $p(z)=p(x)+p(y)$;

добавить z к A.