О сложности расшифровки существенных переменных функции, задающей разбиение булевого куба

Б. В. Воронин, В. В. Осокин

В работе исследуется сложность определения существенных переменных функций из некоторого подкласса класса функций, задающих разбиение булевого куба на подкубы. Показано, что $k \log_2 n$ запросов достаточно для определения всех существенных переменных любой функции из этого подкласса. Получены асимптотики сложностей определения существенных переменных при малых и больших значениях k: $\log_2 n$ и n соответственно.

Введение

Рассматриваемая в работе задача принадлежит к классу задач машинного обучения, в которых требуется за минимальное число запросов к черному ящику получить ту или иную информацию о неизвестной дискретной функции. Под черным ящиком здесь подразумевается некоторый оператор, который «знает» функцию, а под запросом к черному ящику — запрос значения упомянутого оператора на выбранном алгоритмом наборе из области определения функции.

Примером подобной задачи может служить задача расшифровки дискретных функций, в которой требуется полностью восстановить таблицу значений неизвестной функции за минимальное число обращений к черному ящику. Данная задача впервые исследовалась в советской научной школе в середине XX века и рассматривалась для класса монотонных функций. В работе [1] получено точное значение

сложности расшифровки монотонных функций. В работе [2] исследовалась сложность расшифровки пороговых функций. В [3] аналогичная задача решается для функций, задающих разбиение булевого куба на подкубы. Последние 20 лет описанное направление активно развивается и зарубежными учеными. Отправными точками для проводимых за рубежом исследований стали работы [4] и [5].

Задача расшифровки функций является самой сложной из всех задач получения информации о неизвестной функции при помощи упомянутых запросов к черному ящику. Действительно, алгоритм расшифровки должен узнать максимум информации о неизвестной функции — ее таблицу значений. Во многих работах (см., например, [6], [7], [8]) рассматривается задача расшифровки функций в предположении, что не все переменные искомой функции являются существенными. Возникает вопрос связи сложности расшифровки функции со сложностью поиска существенных переменных этой функции. В данной работе исследуется именно задача определения существенных переменных.

Как упоминалось ранее, имеется некоторый оператор, который по набору из области определения функции f выдает значение функции f на этом наборе. Требуется построить алгоритм, который для любой функции f из некоторого класса за минимальное число обращений к этому оператору определяет номера всех существенных переменных функции f.

Для класса функций, задающих разбиение булевого куба на подкубы, известна верхняя оценка сложности полной расшифровки функций из этого класса, равная $k \log n + 2^k$. В настоящей работе исследуется задача определения существенных переменных в некотором подклассе данного класса. Показано, что для любой функции из этого подкласса сложность определения существенных переменных не превосходит $k \log n$. При достаточно больших и достаточно малых k получены асимптотики сложности определения существенных переменных, равные n и $\log n$ соответственно.

Умение быстро находить существенные переменные функции может ускорить ее полную расшифровку. Аналогичная ситуация нередко возникает в теории распознавания образов [9], одной из важных

задач которой является проблема первоначального выбора наиболее информативных признаков и снижения размерности признакового пространства за счет исключения малоинформативных признаков.

Исследованию существенных переменных дискретных функций посвящены также, в частности, работы [10] и [11]. Однако, и в [10], и в [11] задача рассматривается в принципиально иной постановке — таблица значений функции предполагается известной до начала работы каких-либо алгоритмов.

Авторы выражают благодарность профессору Гасанову Э. Э. за постановку задачи и помощь в работе.

1. Постановка задачи и формулировка результатов

Пусть $E_2^n = \{0,1\}^n$ — n-мерный булев куб, $f\colon E_2^n \to \mathbb{Z}_{2^k}$ — некоторая функция от n булевых переменных, k из которых являются существенными. Предполагаем, что функция f при различных фиксациях своих существенных переменных пробегает все значения от 0 до $2^k - 1$.

Обозначим множество всех таких функций через $\Phi^{k,n}$. Через $\mathfrak{F}^{k,n}$ будем обозначать множество алгоритмов, которые для любой $f \in \Phi^{k,n}$ находят номера всех ее существенных переменных при помощи *запросов на значение функции* f. Под запросом понимается набор значений переменных функции, а под ответом на запрос — значение функции на этом наборе.

Алгоритмы в $\mathfrak{F}^{k,n}$ предполагаются условными, то есть при выборе нового запрашиваемого набора они могут учитывать информацию о функции f, полученную из анализа значений f на ранее запрошенных наборах.

Сложностью $\psi(F, f)$ алгоритма $F \in \mathfrak{F}^{k,n}$ на функции $f \in \Phi^{k,n}$ будем называть число запросов на значение функции, необходимое алгоритму F для того, чтобы определить все существенные переменные функции f.

В настоящей работе исследуется величина

$$\tilde{\psi}(k,n) = \min_{F\in\mathfrak{F}^{k,n}} \max_{f\in\Phi^{k,n}} \psi(F,f).$$

Получены следующие асимптотические оценки:

$$\max(k+1,\log_2 n) \lesssim \tilde{\psi}(k,n) \lesssim \min(n, k\log_2 n),$$
$$\tilde{\psi}(k,n) \lesssim \log_2 n + A(k)$$

при $n\to\infty$, где $A(k)$ — некоторая функция, зависящая только от количества существенных переменных. При $k \leqslant \log_2\log_2 n$ выполняется неравенство $A(k) \leqslant 2^k + k^2$.

При определенных k эти оценки становятся асимптотическими равенствами:

Теорема 1. *Пусть* $k = \log_2\log_2 n + \beta(n)$, $2^{\beta(n)} \to 0$ *при* $n\to\infty$. *Тогда*

$$\tilde{\psi}(k,n) \sim \log_2 n.$$

Теорема 2. *При* $k\sim n$, $n\to\infty$ *выполнено*

$$\tilde{\psi}(k,n) \sim n.$$

2. Информационные множества

В данной главе формализуется информация о существенных переменных, доступная алгоритму расшифровки после подачи некоторого числа наборов. Пусть имеется алгоритм $F\in\mathfrak{F}^{k,n}$. На любом шаге мы можем рассматривать множества X^{C} и X^{H} номеров всех известных на данный момент существенных и несущественных переменных соответственно.

С-поднабором набора $\tilde{\alpha} = (\alpha_1,\alpha_2\dots\alpha_n)$ назовем поднабор $\hat{\alpha} = (\alpha_{i_1},\alpha_{i_2}\dots\alpha_{i_s})$, полученный вычеркиванием разрядов с номерами $j, j\in X^{\mathrm{H}}$.

Для двух наборов $\tilde{\alpha}^i = (\alpha_1^i,\alpha_2^i\dots\alpha_n^i)$ и $\tilde{\alpha}^j = (\alpha_1^j,\alpha_2^j\dots\alpha_n^j)$ множество $\hat{X}_{i,j} = \{s \mid \alpha_s^i \neq \alpha_s^j\}$ назовем *информационным*.

Для двух наборов $\tilde{\alpha}^i = (\alpha_1^i, \alpha_2^i \dots \alpha_n^i)$ и $\tilde{\alpha}^j = (\alpha_1^j, \alpha_2^j \dots \alpha_n^j)$ *информационным C-множеством* $X_{i,j}^{\mathrm{C}}$ назовем множество

$$X_{i,j}^{\mathrm{C}} = \left[\begin{array}{l} \hat{X}_{i,j}, f(\tilde{\alpha}^i) \neq f(\tilde{\alpha}^j); \\ \varnothing, f(\tilde{\alpha}^i) = f(\tilde{\alpha}^j). \end{array} \right.$$

Для двух наборов $\tilde{\alpha}^i = (\alpha_1^i, \alpha_2^i \dots \alpha_n^i)$ и $\tilde{\alpha}^j = (\alpha_1^j, \alpha_2^j \dots \alpha_n^j)$ *информационным H-множеством* $X_{i,j}^{\mathrm{H}}$ назовем множество

$$X_{i,j}^{\mathrm{H}} = \left[\begin{array}{l} \hat{X}_{i,j}, f(\tilde{\alpha}^i) = f(\tilde{\alpha}^j); \\ \varnothing, f(\tilde{\alpha}^i) \neq f(\tilde{\alpha}^j). \end{array} \right.$$

Сравнивая два набора, мы получаем информацию либо о существенных, либо о несущественных переменных, в зависимости от значений функции на этих наборах. Информационные C- и H-множества показывают, какую именно информацию мы получаем. Зафиксируем это в виде простого утверждения.

Утверждение 1. *Для любых* i, j *выполнено* $\hat{X}_{i,j} = X_{i,j}^C \cup X_{i,j}^H$, $X_{i,j}^C \cap X_{i,j}^H = \varnothing$.

Рассмотрим информацию о существенности/несущественности некоторой переменной неизвестной функции, доступную алгоритму расшифровки на некотором шаге:

1) мы можем узнать, что переменная является несущественной тогда и только тогда, когда имеется два набора, на которых функция принимает одинаковые значения, отличающихся в разряде, соответствующем этой переменной.
2) мы можем узнать, что переменная является существенной тогда и только тогда, когда имеется два набора, на которых функция принимает разные значения, отличающихся в разряде, соответствующем этой переменной, и при этом про все остальные разряды, в которых наборы отличаются, известно, что они соответствуют несущественным переменным.

Сформулируем эти замечания в более строгой форме.

Утверждение 2. *Для любого номера s выполнено*

- $s \in X^H$ *тогда и только тогда, когда* $\exists i, j$, *такие что* $s \in X^H_{i,j}$.
- $s \in X^C$ *тогда и только тогда, когда* $\exists i, j$, *такие что* $\{s\} \equiv X^C_{i,j} \cap X^H$.

3. Оценка сверху

В данном разделе мы докажем верхнюю оценку сложности расшифровки существенных переменных, верную для любых значений $k \leqslant n$.

Лемма 1. *Для любых k выполнено асимптотическое неравенство* $\tilde{\psi}(k, n) \lesssim k \log_2 n$ *при* $n \to \infty$.

Доказательство. Опишем алгоритм $F \in \mathfrak{F}^{k,n}$, сложность которого на любой функции $f \in \Phi^{k,n}$ не превышает $k \log_2 n$. Для этого на булевом кубе E_2^n выберем произвольную цепочку наборов $y_1, y_2, \ldots, y_{n+1}$, такую что набор y_1 является нулевым, набор y_{n+1} — единичным, а два соседних набора цепочки отличаются лишь по одной переменной. Тогда на этой цепочке функция f будет принимать $k + 1$ значение, причем разные значения на соседних элементах цепи будут означать существенность переменной, по которой они отличаются. Тем самым задача нахождения существенных переменных функции f сведена к нахождению всех пар соседних наборов цепи, значения на которых отличаются.

Доказательство будем проводить индукцией по k. Пусть $k = 1$. Тогда на цепочке f принимает 2 значения — 0 и 1. Запрашиваем значение функции f на нулевом наборе. Далее без ограничения общности считаем, что на нулевом наборе функция принимает значение 0, а на единичном — 1. Необходимо найти номер i такой, что $f(y_i) = 0$, а $f(y_{i+1}) = 1$.

Запросим значение f на среднем наборе цепи y_i, где $i = \left[\frac{n+2}{2}\right]$. Мы получим одно из двух значений — 0 или 1. Так как это значение совпадает с одним из значений на конечном элементе цепи, это будет

означать, что между y_i и соответствующим элементом все элементы цепи принимают одинаковое значение. Тем самым, мы разбили цепь на две подцепи (равной или примерно равной длины), и в дальнейшем можем рассматривать только одну половину. Выберем центральный элемент и повторим наши действия. В итоге мы получим два соседних набора, значения на которых будут отличаться, тем самым найдя существенную переменную.

Теперь оценим время работы такого алгоритма. Положим $r = 2^{]\log_2 n[}$. Тогда $r > n$ и очевидно, что подобный алгоритм будет работать на функциях от r переменных не меньше, чем на фунциях от n. Будем считать, что значения на средних наборах подцепей всегда равно 1, то есть мы всегда рассматриваем подцепи, начинающиеся с y_1. Тогда последовательность номеров этих переменных будет выглядеть следующим образом:

$$\frac{r}{2}, \frac{r}{4}, \ldots, \frac{r}{2^{]\log_2 n[-1}} = 2.$$

Учитывая дополнительный запрос $f(0, 0, \ldots, 0)$, чтобы узнать значение на одном конце цепи (значение на другом будет отличаться, а значит — однозначно определено), получаем время работы алгоритма:

$$\psi(F, f) = 1 +]\log_2 n[- 1 =]\log_2 n[.$$

Пусть утверждение леммы выполнено для $k = s$. Докажем его для $k = s + 1$. Спросим значения f на конечных наборах цепи и на среднем. Если значение на среднем наборе совпадает с одним из значений на конечных наборах, то продолжим алгоритм, аналогичный алгоритму для $k = 1$, пока не появится подцепь, где значение на среднем элементе отлично от значений на конечных. Пусть к этому моменту мы сделали $2 + l$ запросов. Тем самым, последний средний элемент разобьет подцепь на две, каждая длиной не больше чем из $n' = \frac{n}{2^l}$ элементов. На этих подцепях функция меняет значения x и $k - x$ раз, соответствуя x и $k - x$ существенным переменным. Время работы алгоритма для каждой из подцепей не превосходит $x\,]\log_2 n'[$ и $(k - x)\,]\log_2 n'[$. Тогда сложность алгоритма F на функции f равна

$$\psi(F,f) = 2 + l + x\,\big]\log_2 n'\big[+ (k-x)\,\big]\log_2 n'\big[= \\ = 2 + l + k\,\Big]\log_2 \frac{n}{2^l}\Big[= 2 + l + k\,]\log_2 n[- kl \sim k\log_2 n.$$

Этим мы заканчиваем доказательство леммы.

Лемма 2. *Для любых k, n выполнено $\tilde{\psi}(k,n) \leqslant n$.*

Доказательство. Построим алгоритм, сложность которого не превосходит n. Рассмотрим последовательность наборов

$$\tilde{\alpha}_1 = (1,0,0,\ldots,0,0);$$
$$\tilde{\alpha}_2 = (0,1,0,\ldots,0,0);$$
$$\vdots$$
$$\tilde{\alpha}_{n-1} = (0,0,0,\ldots,1,0);$$
$$\tilde{\alpha}_n = (0,0,0,\ldots,0,0).$$

и будем сравнивать значения всех наборов с последним. Совпадение значений i-го и n-го набора означает, что найдена несущественная переменная. В итоге возможны два варианта:

1) найдено $n-k$ несущественных переменных,
2) найдена $n-k-1$ несущественная переменная. Тогда последняя переменная тоже будет несущественной.

Остальные k переменных являются существенными. Лемма доказана.

Теорема 3. *Для любых k при $n \to \infty$ выполнено асимптотическое неравенство*

$$\tilde{\psi}(k,n) \lesssim \min(n, k\log_2 n).$$

4. Оценка снизу

В данном разделе мы докажем нижнюю оценку сложности расшифровки существенных переменных. Как и в предыдущем разделе, полученный результат верен для любых значений $k \leqslant n$.

Приводимую далее лемму докажем для $k \leqslant \frac{n}{2}$. Для остальных k лемма 4 даст более сильную оценку.

Лемма 3. *Если* $k \leqslant \frac{n}{2}$, *то* $\tilde{\psi}(k,n) \geqslant \log_2 n - 1$.

Доказательство. Положим значения на нулевом и единичном наборах равными 0 и $2^k - 1$ соответственно. Сразу будем считать, что последние $k-1$ переменные являются существенными, а еще одну k-ую существенную переменную будем выбирать исходя из запросов алгоритма.

Вместо наборов $\tilde{\alpha} = (\alpha_1, \alpha_2 \dots \alpha_n)$ мы будем по сути рассматривать поднаборы $\tilde{\alpha}' = (\alpha_1, \alpha_2 \dots \alpha_{n-k+1})$, а также $\tilde{\alpha}^0 = (0, 0 \dots 0, \alpha_{n-k+2} \dots \alpha_n)$ и $\tilde{\alpha}^1 = (1, 1 \dots 1, \alpha_{n-k+2} \dots \alpha_n)$, значения на которых выберем заранее. Для каждого такого поднабора рассматриваем его вес $w(\tilde{\alpha}')$ и сравниваем с половиной длины поднабора — $\left[\frac{n-k+1}{2}\right]$. Если вес поднабора больше, то есть единиц в нем больше, чем нулей, то определяем значение на наборе $\tilde{\alpha}$ равным значению на наборе $\tilde{\alpha}^1$. Иначе — значению на наборе $\tilde{\alpha}^0$.

Тем самым, поднабор разобьется на два поднабора — состоящих из одних нулей и из одних единиц. Все разряды, относящиеся к более короткому, будут соответствовать несущественным переменным, а значит на следующем шаге можно будет рассматривать более длинный поднабор. Так длина будет постепенно уменьшаться, но на i-ом шаге длина выбранного поднабора будет не меньше, чем

$$\left[\frac{n-k+1}{2^i}\right] \geqslant \left[\frac{n}{2^{i+1}}\right].$$

В итоге мы должны получить поднабор длины 1, а значит для сложности алгоритма будет выполнено

$$\psi(F, f) \geqslant \log_2 n - 1.$$

Тем самым, лемма доказана.

Утверждение 3. *Для любого* n *выполнено* $\tilde{\psi}(n-1, n) \geqslant n$.

Доказательство. Пусть наш алгоритм подал $n-1$ различный запрос $\tilde{\alpha}^1, \dots, \tilde{\alpha}^{n-1}$. На каждом запросе будем давать новое значение (поскольку всего их $2^k = 2^{n-1}$, то проблем с нехваткой значений не

будет). Построим матрицу $M_{(n-1)\times n}$, строками которой будут являться наборы $\tilde{\alpha}^1, \ldots, \tilde{\alpha}^{n-1}$:

$$M = \begin{pmatrix} \alpha_1^1 & \alpha_2^1 & \ldots & \alpha_n^1 \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{n-1} & \alpha_2^{n-1} & \ldots & \alpha_n^{n-1} \end{pmatrix}.$$

Обозначим столбцы матрицы через $\tilde{\beta}^i$. При этом можно считать, что $\nexists i : \tilde{\beta}^i = 0$. В противном случае мы заменили бы все нулевые столбцы на единичные, сохранив при этом информационные множества.

Заметим, что $rk(M) \leqslant n-1$. Значит, $\exists i_1 \ldots i_t : \sum_{j=1}^{t} \tilde{\beta}^{i_j} = 0$, причем $t \geqslant 2$. Для каждой переменной i_j из этой формулы следует, что соответствующий столбец выражается через сумму других, а значит, не существует двух наборов, отличающихся только по i_j-ой переменной. В связи с тем, что согласно утверждению 2 нам доступна лишь C-информация, нельзя однозначно определить существенность переменной с номером i_j ни для какого i_j. А так как $t \geqslant 2$, то подав $n-1$ запрос, алгоритм позволит узнать не более, чем $n-2$ существенные переменные. Утверждение доказано.

Из леммы 2 и утверждения 3 получаем следующий результат:

Утверждение 4. *Для любого n выполнено $\tilde{\psi}(n-1, n) = n$.*

Из него можно получить следующую теорему:

Лемма 4. *Для любых k и n выполнено*

$$\tilde{\psi}(k, n) \geqslant k+1.$$

Доказательство. Будем считать, что последние $n-k-1$ переменных являются несущественными, и в зависимости от этого подавать разные значения на запросы алгоритма. Если поднабор нового запроса, состоящий из первых $k+1$ переменных, не совпадает ни с одним из поднаборов старых запросов, даем новое значение. Иначе даем значение, равное значению на запросе с совпадающим поднабором.

По сути, задача сводится к нахождению k существенных переменных из $k+1$, сложность которой по утверждению 4 равна $k+1$. Это завершает доказательство леммы.

Из лемм 3 и 4 получаем следующую теорему.

Теорема 4. *Для любых $k \leqslant n$ выполнено неравенство*

$$\tilde{\psi}(k,n) \geqslant \max(k+1, \log_2 n - 1).$$

5. Оценка сверху для случая малого количества существенных переменных

В предыдущих разделах рассматривались оценки сложности расшифровки существенных переменных, верные при любых k. В настоящем разделе для малых k мы построим алгоритм близкий к оптимальному.

Начнем с рассмотрения частных случаев $k = 2$ и $k = 3$.

Утверждение 5. *При $k = 2$ и любом n выполнено*

$$\tilde{\psi}(2,n) \leqslant]\log_2 n[+ 2.$$

Доказательство. Построим алгоритм, который для любой функции f будет находить ее существенные переменные за указанное время.

Сначала узнаем значения на нулевом и единичном наборах. После этого построим серию из t наборов $\tilde{\alpha}^1 \dots \tilde{\alpha}^t$ обладающую следующим свойством: если представить эти наборы как матрицу M размера $t \times n$, то все ее стобцы будут различны, и разобьются на пары противоположных за исключением, быть может, одного. Для этого должно выполняться $t \geqslant \log_2 n$. Возьмем t равным $]\log_2 n[$.

Если мы будем знать соответствие между значениями функции и поднаборами существенных переменных, то нам будет достаточно составить матрицу из поднаборов существенных переменных, соответствующих значениям функции на наборах $\tilde{\alpha}^1 \dots \tilde{\alpha}^t$, а затем найти в матрице M столбцы, равные столбцам этой матрицы, которые и будут соответствовать существенным переменным в силу вышеуказанного свойства матрицы M.

Рассмотрим булев куб E_2^2. Он состоит из трех слоев, где под слоем понимается множество наборов одинакового веса. Первый и третий слои состоят из одного элемента каждый, значения на которых мы узнаем после первых двух запросов. Будем считать, что эти значения равны 0 и 3. Оставшиеся два значения функция принимает на двух элементах среднего слоя. При этом оба варианта соответствия значений элементам приведут нас к одинаковым существенным переменным, так как при их нахождении поменяется лишь порядок столбцов в матрице, соответствующей столбцу значений, сами же столбцы будут совпадать в обоих вариантах. Тем самым мы можем положить, что набору $(0,1)$ соответствует значение 1, а набору $(1,0)$ — 2.

Это означает, что подав $]\log_2 n[$ наборов плюс 2 дополнительных (нулевой и единичный), мы однозначно определим существенные переменные. Утверждение доказано.

Утверждение 6. *При $k = 3$ и любом n выполнено:*

$$\tilde{\psi}(3, n) \leqslant]\log_2 n[+ 6.$$

Доказательство. Рассмотрим алгоритм из доказательства утверждения 5. В случае трех существенных переменных мы имеем дело с кубом E_2^3, у которого четыре слоя, причем два средних имеют по 3 элемента — всего им соответствуют 6 значений от 1 до 6. При этом различные варианты соответствия наборов значениям дают различные столбцы, поэтому просто серии из $]\log_2 n[$ запросов будет недостаточно.

Изменим наш алгоритм следующим образом. Каждый раз, когда после очередного запроса $\tilde{\alpha}^i$ мы получим новое, ранее не возникавшее, значение, спросим значение на отрицании этого набора. Тем самым все значения разобьются на пары значений, соответствующих противоположным наборам. Получив две такие пары (например $1 \leftrightarrow 6$, $2 \leftrightarrow 5$), последняя третья пара определится однозначно $(3 \leftrightarrow 4)$.

После этого возьмем два любых набора, на которых функция принимает различные значения (причем не соответствующих противоположным наборам и нулевому или единичному наборам) и спросим

значение на дизъюнкции этих наборов. Рассмотрим всевозможные варианты:

$\tilde{\alpha}$	$\tilde{\beta}$	$\tilde{\alpha} \vee \tilde{\beta}$
1 $(1,0,0)$	1 $(0,1,0)$	2 $(1,1,0)$
1 $(1,0,0)$	2 $(1,1,0)$	2 $(1,1,0) = \tilde{\beta}$
2 $(1,0,1)$	2 $(1,1,0)$	3 $(1,1,1)$

Первые цифры в таблице — номера слоев, номер слоя равен весу наборов этого слоя.

Тем самым, по значению на новом наборе мы однозначно определим принадлежность к слоям наборов $\tilde{\alpha}$ и $\tilde{\beta}$, а значит и противоположных им. Взяв два других набора, значение одного из которых будет равно двум из оставшихся значений, мы установим полное соответствие между значениями и слоями.

Как и в случае $k = 2$, мы можем произвольно распределить значения по первому слою, так как это будет влиять лишь на перестановки столбцов матрицы.

Подсчитаем суммарную сложность описанного алгоритма:

$$\psi(F, f) = 2 +]\log_2 n[+ 2 + 2 =]\log_2 n[+ 6.$$

Утверждение доказано.

Заметим, что если среди значений на $]\log_2 n[$ наборах встречаются не все возможные, то не имеет значения соответствие этих значений наборам, так как соответствующие им строки все равно не будут встречаться в матрице, из которой мы находим существенные переменные. Заметим также, что вместо дизъюнкции можно взять конъюнкцию и провести аналогичные рассуждения.

Для общего случая верна следующая оценка сверху:

Теорема 5. *Существует функция $A(k)$ такая, что для любых n*

$$\tilde{\psi}(k, n) \leqslant \log_2 n + A(k),$$

причем функция $A(k)$ зависит только от количества существенных переменных и при $k < \log_2 \log_2 n$ удовлетворяет неравенству

$$A(k) \leqslant 2^k + k^2.$$

Доказательство. Для начала докажем первую часть теоремы. Возьмем алгоритм из утверждений 5 и 6 и подсчитаем количество вариантов соответствия значений функции поднаборам существенных переменных.

Количество таких вариантов будет зависить только от k, но не от n. Тем самым у нас получается определенное число вариантов «возможных существенных переменных» — число всех таких переменных тоже будет зависить только от k. Каждую такую переменную проверяем на существенность, спрашивая значения на наборе, где в разряде, соответствующем этой переменной, стоит 1, а в остальных — 0, и сравнивая значения функции на этом наборе со значением на нулевом наборе. Если значения на этих двух наборах совпадают, то переменная не является существенной, в противном случае, переменная является существенной.

Так как один из вариантов соответствия является верным, «реальные» существенные переменные содержатся в множестве «возможных», а значит в конечном счете алгоритм выполнит свою задачу, причем время его работы будет иметь вид $\log_2 n + A(k)$.

Перейдем к оценке функции $A(k)$ в случае $k < \log_2 \log_2 n$. Можем считать n степенью двойки — в противном случае добавим несущественных переменных. Возьмем «стандартные» $\log_2 n$ наборов (см. выше). Пусть на этих наборах принимается всего s различных значений ($s \leqslant 2^k$). Так как $2^k < \log_2 n$, то некоторые значения будут приниматься несколько раз. Возьмем два набора $\tilde{\alpha}^i$ и $\tilde{\alpha}^j$, значения на которых совпадают, и укоротим все наборы, вычеркнув все разряды, в которых данные наборы различаются, так как переменные, им соответствующие, будут заведомо несущественными. При этом наборы $\tilde{\alpha}^i$ и $\tilde{\alpha}^j$ и только они склеятся в один. Так будем продолжать до тех пор, пока у нас не окажется s наборов с различными значениями.

Оценим длину полученных наборов.

Структура системы из $\log_2 n$ наборов такова, что l-ый набор представляет из себя массивы из $2^{\log_2 n - l}$ нулей и единиц, идущих попеременно. Наборы $\tilde{\alpha}^i$ и $\tilde{\alpha}^j$ отличаются ровно в половине разрядов, значит количество разрядов — кандидатов в существенные переменные — при вычеркивании различающихся разрядов сократится вдвое.

Если рассматривать наборы как матрицу, то можно выделить в ней 4 блока (подматрицы) в зависимости от значений переменных в этих двух наборах — соответственно $(0,0)$ (то есть столбцы, в которых на i-ом и j-ом местах стоят нули), $(0,1)$, $(1,0)$, $(1,1)$. После вычеркивания переменных, вычеркнутся целиком блоки $(0,1)$ и $(1,0)$. Теперь возьмем набор $\tilde{\alpha}^l$ и рассмотрим 3 варианта:

1) $l > j$. Для таких наборов структура каждого блока является такой же, как и для системы всех наборов. Поэтому длины массивов не изменятся.
2) $j > l > i$. Длина каждого массива уменьшится вдвое, так как один такой массив содержится в равном числе блоков $(0,0)$ и $(0,1)$.
3) $i > l$. Как и во втором варианте, за счет равного числа блоков, в котором содержится один массив, длина массивов уменьшится вдвое.

Например, в случае

$$(0,0,0,0,1,1,1,1)$$
$$(0,0,1,1,0,0,1,1)$$
$$(0,1,0,1,0,1,0,1)$$

варианты 1, 2 и 3 соответствуют тройкам наборов (i,j,l), (i,l,j) и (l,i,j) соответственно. Длина массивов i-го набора сократится вдвое.

Тем самым после сравнения двух наборов, принимающих одинаковые значения, у нас останется на один набор меньше, причем эти наборы будут в два раза короче и их структура останется прежней. В дальнейшем под $\tilde{\alpha}^i$ мы будем каждый раз подразумевать уже укороченные наборы, а не наборы первоначальной длины. В итоге после всех таких операций у нас будет s наборов длины 2^s.

Если $s \leqslant \frac{2^k}{k}$, то по лемме 1 имеем

$$\tilde{\psi}(k,n) \leqslant \log_2 n + k \log_2 2^{\frac{2^k}{k}} = \log_2 n + k\frac{2^k}{k} = \log_2 n + 2^k,$$

тем самым утверждение теоремы выполнено. Иначе будем рассматривать всевозможные суммы $\tilde{\beta} = \tilde{\alpha}^i \oplus \tilde{\alpha}^j$ и дописывать их к множеству

уже имеющихся наборов. При этом все наборы будем подразделять на два подмножества: $\log_2 L$ наборов (где L — текущая их длина), соответствующих первоначальной структуре (будем называть «структурными») и остальные наборы, из этой структуры выбивающиеся. Возможны 3 варианта:

1) $\tilde{\beta}$ уже присутствует среди наборов. Тогда мы переходим к другой паре наборов. Так как мы не запрашиваем значение функции на наборе в данном случае, время работы алгоритма остается неизменным.
2) $\tilde{\beta}$ не присутствует среди наборов, и его значение не совпадает ни с одним из уже полученных значений. Тогда добавим набор в множество к остальным и перейдем к следующей паре наборов. Всего за счет таких наборов ко времени работы алгоритма прибавится не более чем $2^k - s$.
3) $\tilde{\beta}$ не присутствует среди наборов, и его значение совпадает с каким-то из наборов $\tilde{\gamma}$. Вычеркнем все переменные, соответствующие разрядам, в которых эти два набора различаются. Покажем, что тогда наша структура сохранится.

Для начала покажем, что наборы $\tilde{\beta}$ и $\tilde{\gamma}$ различаются ровно в половине своих переменных. Каждый из них является суммой нескольких структурных наборов. Рассмотрим сумму $\tilde{\beta} \oplus \tilde{\gamma}$, которая также является суммой нескольких структурных наборов. Количество разрядов, где наборы различаются — вес их суммы.

Структурные наборы устроены таким образом, что в разрядах, симметричных относительно середины, находятся противоположные значения. Поэтому если $\tilde{\beta} \oplus \tilde{\gamma}$ представляет из себя сумму нечетного числа наборов, то это свойство сохранится и для него:

$$\alpha_t^{i_1} \oplus \alpha_t^{i_2} \oplus \ldots \oplus \alpha_t^{i_{2r+1}} = (\alpha_{L-t}^{i_1} \oplus 1) \oplus (\alpha_{L-t}^{i_2} \oplus 1) \oplus \ldots \oplus (\alpha_{L-t}^{i_{2r+1}} \oplus 1) = \\ = \alpha_{L-t}^{i_1} \oplus \alpha_{L-t}^{i_2} \oplus \ldots \oplus \alpha_{L-t}^{i_{2r+1}} \oplus 1,$$

тем самым нулей и единиц в этом наборе будет поровну. Если количество слагаемых четно, то сумму можно представить как $\tilde{\alpha}^i \oplus \tilde{\delta}$, где

$\tilde{\alpha}^i$ — один из структурных наборов, а $\tilde{\tilde{\delta}}$ — сумма всех остальных. Будем считать, что $i = 1$, то есть это первый из структурных наборов (в противном случае мы может переставить строки всех наборов нужным образом — на количество нулей и единиц это не повлияет). Этот набор разобьет все остальные структурные на блоки, соответствующие 0 и 1 в разрядах $\tilde{\alpha}^i$. Тогда в каждом блоке количество нулей и единиц в наборе $\tilde{\tilde{\delta}}$ будет совпадать. А значит и в сумме нулей и единиц будет поровну.

Итак, мы установили, что всегда будем иметь дело с наборами, в которых поровну нулей и единиц. Единственное исключение — нулевой набор, который получится как сумма $\tilde{\alpha}^i \oplus \tilde{\alpha}^i$.

Теперь осталось убедиться, что структура всегда будет сохраняться. Считаем, что у нас уже выбраны два набора и получено соотношение $f(\tilde{\tilde{\beta}} \oplus \tilde{\gamma}) = f(\tilde{\tilde{\delta}})$. Каждый из этих наборов является либо сам по себе структурным, либо суммой некоторых структурных наборов, уже имеющихся в нашем множестве наборов. Среди них есть набор с наибольшим номером, то есть с наименьшей длинной массива. Рассмотрим один из таких массивов и 2 блока, ему соответствующие — для разрядов с нулями и с единицами. Эти блоки будут совпадать, а значит при сравнении $\tilde{\tilde{\beta}} \oplus \tilde{\gamma}$ и $\tilde{\tilde{\delta}}$ будет вычеркиваться один из таких блоков для каждого массива. Для наборов с большим номером, то есть лежащих ниже, длина массивов останется прежней. Для наборов с меньшим она соответственно будет укорачиваться в некоторое число раз, но при этом все длины массивов будут охвачены. То есть после вычеркивания несущественных переменных, структура сохранится.

Так мы будем действовать, пока в итоге не придем к множеству таких наборов, что сумма любых двух уже будет входить в это множество. Когда такое произойдет, будем находить существенные переменные из оставшихся переменных по алгоритму из леммы 1.

Оценим, сколько переменных у нас останется. В начале имеем 2^s переменных и s структурных наборов. При появлении нового, ранее не встречавшегося, значения на сумме наборов, добавляется новый набор. Количество таких добавленных наборов не превышает $2^k - s$. При каждом «удачном» выборе двух наборов для суммирования ко-

личество переменных сокращается вдвое, количество структурных наборов уменьшается на один, а количество всех наборов остается неизменным.

Пусть количество переменных сократилось вдвое T раз. Тогда количество переменных равно 2^{s-T}, количество структурных наборов — $s-T$. Количество всех наборов не превосходит $s+(2^k-s)=2^k$. На каком-то шаге алгоритма получим, что сумма любых двух наборов нашего множества тоже лежит в нем, то есть любой набор

$$\tilde{\alpha}=\varepsilon_1\tilde{\alpha}^1\oplus\varepsilon_2\tilde{\alpha}^2\oplus\ldots\oplus\varepsilon_{s-T}\tilde{\alpha}^{s-T},$$

где $\varepsilon_i\in\{0,1\}$ принадлежит этому множеству.

Всего таких наборов 2^{s-T}, а значит должно выполняться неравенство

$$2^{s-T}\leqslant 2^k\Leftrightarrow s-T\leqslant k\Rightarrow s-k\leqslant T\leqslant s.$$

Оценим $\tilde{\psi}(k,n)$:

$$\tilde{\psi}(k,n)\leqslant\log_2 n+(2^k-s)+T+k\log_2 2^{s-T}\leqslant\log_2 n+2^k+k^2.$$

Этим неравенством мы заканчиваем доказательство теоремы.

6. Асимптотика сложности расшифровки в частных случаях

Докажем теорему 1.

Доказательство. Из теоремы 5 следует асимптотическое неравенство $\tilde{\psi}(k,n)\lesssim\log_2 n+2^k$. В силу того, что $\tilde{\psi}(k,n)\geqslant\log_2 n$, асимптотика будет тогда, когда второе слагаемое мало по сравнению с первым:

$$2^k\ll\log_2 n;$$

$$2^k=\alpha(n)\log_2 n,\text{ где }\alpha(n)\to 0\text{ при }n\to\infty;$$

$$k=\log_2\left(\alpha(n)\log_2 n\right)=\log_2\log_2 n+\log_2\alpha(n);$$

$$\log_2\frac{2^k}{\alpha(n)}=\log_2\log_2 n,$$

а значит, надо взять $k = \log_2 \log_2 n + \beta(n)$, где функция $\beta(n)$ такая, что $2^{\beta(n)}$ — бесконечно малая. Теорема доказана.

Заметим, что $2^{\beta(n)} \to 0 \Leftrightarrow \beta(n) \to -\infty$. Тем самым, в качестве $\beta(n)$ можно взять любую функцию, стремящуюся к минус бесконечности сколь угодно медленно. Например, подойдет $\beta(n) = -\log_2 \log_2 \log_2 \log_2 n$.

Утверждение теоремы 2 сразу следует из теорем 3 и 4.

Список литературы

[1] Hansel G. О числе монотонных булевых функций n переменных // Кибернетич. сб. Новая серия. Вып. 5. М.: Мир, 1968.

[2] Золотых Н. Ю. Расшифровка пороговых и близких к ним функций многозначной логики / Диссертация на соискание степени кандидата физико-математических наук. Нижегородский государственный университет, 1998.

[3] Осокин В. В. Асимптотически оптимальный алгоритм расшифровки разбиения булевого куба на подкубы // Интеллектуальные системы. Т. 11. 2007. С. 587–606.

[4] Angluin D. Queries and Concept Learning // Machine Learning. Vol. 2. 1988. P. 319–342.

[5] Valiant L. A Theory of the Learnable // Comm. ACM. 27. 1984. P. 1134–1142.

[6] Uehara R., Tsuchida K., Wegener I. Optimal attribute-efficient learning of disjunction, parity, and threshold functions // Lecture Notes In Computer Science. Vol. 1208. 1997.

[7] Damaschke P. Adaptive Versus Nonadaptive Attribute-Efficient Learning // Machine Learning. 41. 2000. P. 197–215.

[8] Осокин В. В. О сложности расшифровки разбиения булевого куба на подкубы // Дискретная математика. Т. 20, вып. 2. 2008. С. 46–62.

[9] Кудрявцев В. Б., Андреев А. Е., Гасанов Э. Э. Теория тестового распознавания. М.: ФИЗМАТЛИТ, 2007.

[10] Журавлев Ю. И. О несущественных переменных не всюду определенных функций алгебры логики // Дискретный анализ. Сборник трудов. Новосибирск, 1963.

[11] Вороненко А. А. Методы представления дискретных функций в задачах подсчета, тестирования и распознавания свойств / Диссертация на соискание ученой степени доктора физико-математических наук. Москва, 2007.