ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ №4(35), 2017, с. 133–147

УДК 004.021

А. И. Мамонтов, С. М. Рябинов

Об одном методе экономии памяти при классификации текстов

Аннотация. В статье исследуется метод экономии памяти в задачах классификации текстов поиском совпадающих частей линейных полиномов. В начале приводится алгоритм поиска совпадающих частей в линейных полиномах с целыми коэффициентами. Этот алгоритм позволяет вычислять системы линейных полиномов с целыми коэффициентами быстрее и использовать для их хранения меньше памяти. Затем алгоритм применяется для поиска совпадающих частей линейных полиномов, возникающих при классификации текстов с помощью байесовского классификатора, и приводятся вычислительные эксперименты, демонстрирующие экономию памяти.

Ключевые слова и фразы: классификация текстов, линейные полиномы, целые числа, байесовский классификатор.

Введение

Успешность практического применения того или иного алгоритма зависит от таких важных характеристик, как скорость его работы и объём памяти, необходимый для его реализации. Поэтому вопросы, связанные с быстрым вычислением полиномов с различными коэффициентами и уменьшением сложности полиномиальных представлений, достаточно активно исследуются (см., например, [1–5]). В нашей работе изучается метод организации эффективного вычисления системы линейных полиномов с целыми коэффициентами, основанный на поиске совпадающих частей в линейных полиномах, то есть поиске подформул. Отметим, что подходы, основанные на поиске подформул встречаются при организации вычислений и представлении знаний [6].

В нашей работе полиномы представляются в виде композиций (суперпозиций) своих частей. Подобные исследования проведены, например, в [7–9]. В работах [10, 11] суперпозиции полиномов применяются для конструкций, встречающихся при классификации, — нейронных

Работа выполнена при поддержке Российского фонда фундаментальных исследований (грант № 17-01-00485а).

DOI: 10.25209/2079-3316-2017-8-4-133-147

сетей. Отметим, что иногда в области представления полиномов с помощью суперпозиций других полиномов встречаются трудные задачи. Например, задача о возможности получить с помощью системы полиномов с целыми коэффициентами и операции суперпозиции всех полиномов с целыми коэффициентами алгоритмически неразрешима [7].

Частью математического аппарата байесовского классификатора являются взвешенные суммы значений признаков — линейные разделяющие функции, для работы с этими функциями мы применяем алгоритм, использующий суперпозиции.

При множественной классификации текстовых документов с помощью байесовского классификатора возникает достаточно много похожих признаков — словосочетаний с похожими вещественными числовыми характеристиками. Мы округляем эти характеристики до целых чисел, осуществляем поиск совпадающих характеристик и их эффективное хранение и вычисление.

Мы не только применяем способ эффективного вычисления систем линейных полиномов, который был нами описан в [12], но и развиваем его в настоящей работе. Нами описан алгоритм поиска совпадающих частей в линейных полиномах, вычислена его оценка сложности, показано, как можно использовать линейные полиномы с целыми коэффициентами в байесовском классификаторе текстовых документов, проведены вычислительные эксперименты, показывающие эффективность модели на конкретных примерах.

Основной рассматриваемый нами вопрос — экономия памяти при сохранении качества классификации. Вопрос улучшения качества классификации нами не рассматривается.

Таким образом, при экспериментах нас интересуют:

1. Размер базы данных.
2. Сохранение качества классификации.

1. Алгоритм поиска совпадающих частей линейных полиномов

1.1. Постановка задачи

Дана система линейных полиномов:

$$P_1(x_1, \ldots, x_n) = a_{01} + \sum_{i=1}^{n} a_{i1} x_i,$$

$$\ldots$$

$$P_m(x_1, \ldots, x_n) = a_{0m} + \sum_{i=1}^{n} a_{im} x_i,$$

где a_{ij} - целые числа.

При этом $|a_{ij}| < A$, где A — некоторое небольшое положительное целое число.

Требуется многократно вычислять значения этой системы полиномов при различных значениях $(x_1, \ldots, x_n)$, а также экономно хранить эту систему в базе данных.

1.2. Совпадающие части

Пусть у двух полиномов есть совпадающая часть, то есть при некоторых различных $j_1,\ j_2 \in \{1, \ldots, m\}$ и некоторых различных $i_1, \ldots, i_k \in \{1, \ldots, n\}$ выполняются равенства $a_{i_1 j_1} = a_{i_1 j_2}, \ldots\ a_{i_k j_1} = a_{i_k j_2}$. Обозначим

$$P(x_1, \ldots, x_n) = \sum_{t=1}^{k} a_{i_t j_1} x_{i_t},$$

$$R_{j_1}(x_1, \ldots, x_n) = P_{j_1}(x_1, \ldots, x_n) - P(x_1, \ldots, x_n),$$
$$R_{j_2}(x_1, \ldots, x_n) = P_{j_2}(x_1, \ldots, x_n) - P(x_1, \ldots, x_n).$$

Следует отметить, что $R_{j_1}(x_1, \ldots, x_n)$ и $R_{j_2}(x_1, \ldots, x_n)$ - по построению линейные полиномы, тогда

$$P_{j_1}(x_1, \ldots, x_n) = R_{j_1}(x_1, \ldots, x_n) + P(x_1, \ldots, x_n),$$
$$P_{j_2}(x_1, \ldots, x_n) = R_{j_2}(x_1, \ldots, x_n) + P(x_1, \ldots, x_n).$$

Предлагается сначала вычислять $P(x_1, \ldots, x_n)$, а затем $R_{j_1}(x_1, \ldots, x_n)$, $R_{j_2}(x_1, \ldots, x_n)$ и $P_{j_1}(x_1, \ldots, x_n)$, $P_{j_2}(x_1, \ldots, x_n)$. В этом случае объем вычислений сокращается на количество операций в полиноме P, т.е. на $k-1$ сложений и k умножений. Количество ненулевых коэффициентов в полиномах P, R_{j_1}, R_{j_2} меньше, чем количество ненулевых коэффициентов в полиномах P_{j_1}, P_{j_2} на количество ненулевых коэффициентов в полиноме P.

1.3. Поиск совпадающих частей линейных полиномов

Для различных $j_1, j_2, \ldots, j_t \in \{1, \ldots, m\}$ обозначим через $S_{j_1 j_2 \ldots j_t}$ сумму всех различных мономов, входящих в совпадающую часть полиномов с номерами $j_1, j_2, \ldots, j_t$ и не входящих ни в какую совпадающую часть этих полиномов с номерами $j_1, j_2, \ldots, j_t$ и какого-нибудь ещё полинома. То есть в $S_{j_1 j_2 \ldots j_t}$ входят только те мономы, которые входят

также в каждый из полиномов с номерами $j_1, j_2, \ldots, j_t$, и отсутствуют в остальных полиномах системы.

Далее можно последовательно найти все $S_{j_1 j_2 \ldots j_t}$, например, с помощью следующего **алгоритма** S:

1. Для каждой переменной x_i, $i = 1, \ldots, n$:

1.1. Представим значения коэффициентов полиномов при x_i как отдельный двумерный массив с элементами $\langle$номер полинома j, значение коэффициента при $x_i\rangle$.

1.2. Отсортируем полученный в пункте 1.1 массив по значениям коэффициентов при x_i.

1.3. В цикле проходим отсортированный массив и все ненулевые коэффициенты заносим в какой-то из $S_{j_1 j_2 \ldots j_t}$, при этом нужно постепенно сформировывать индекс $j_1 j_2 \ldots j_t$.

Пример 1. Дана система линейных полиномов:

$$\begin{aligned} P_1(x_1, \ldots, x_7) =& x_1 + 2x_5 + 3x_6 + 2x_7, \\ P_2(x_1, \ldots, x_7) =& 2x_1 + 4x_2 + 2x_3 + 2x_5, \\ P_3(x_1, \ldots, x_7) =& 4x_2 + 3x_6, \\ P_4(x_1, \ldots, x_7) =& 2x_1 + 4x_2 + 2x_3 + 2x_5. \end{aligned}$$

Найдём для этой системы все $S_{j_1 j_2 \ldots j_t}$:

1.1. Представим значения всех коэффициентов при x_1 как двумерный массив: $\langle 1, 1\rangle, \langle 2, 2\rangle, \langle 3, 0\rangle, \langle 4, 2\rangle$.

1.2. Отсортируем полученный в пункте 1.1 массив по значениям коэффициентов при x_1: $\langle 3, 0\rangle, \langle 1, 1\rangle, \langle 2, 2\rangle, \langle 4, 2\rangle$.

1.3. Проходим отсортированный массив и формируем $S_{j_1 j_2 \ldots j_t}$. Первым встретился элемент $\langle 3, 0\rangle$. Коэффициент равен 0, поэтому пропускаем его.

Следующий элемент $\langle 1, 1\rangle$, коэффициент равен 1. Начинаем формировать индекс $j_1 j_2 \ldots j_t$. Номер полинома 1, так что предварительно индекс равен 1. Так как в следующем элементе коэффициент $2 > 1$, то коэффициенты с 1 мы больше не встретим, то есть индекс для 1 сформирован, добавляем x_1 в $S_1(x_1, \ldots, x_7)$.

Следующий элемент $\langle 2, 2\rangle$. Коэффициент равен 2. Начинаем формировать следующий индекс $j_1 j_2 \ldots j_t$ для коэффициента 2. Номер полинома 2, так что предварительно индекс равен 2. Следующий элемент $\langle 4, 2\rangle$. Коэффициент опять равен 2, а номер полинома 4. Индекс становится равен 24, этот элемент последний, поэтому добавляем $2x_1$ в $S_{24}(x_1, \ldots, x_7)$.

Продолжая эти действия для следующих переменных, получаем ответ:

$$\begin{aligned} S_1(x_1,\ldots,x_7) &= x_1 + 2x_7,\\ S_{124}(x_1,\ldots,x_7) &= 2x_5,\\ S_{13}(x_1,\ldots,x_7) &= 3x_6,\\ S_{234}(x_1,\ldots,x_7) &= 4x_2,\\ S_{24}(x_1,\ldots,x_7) &= 2x_1 + 2x_3. \end{aligned}$$

Теорема. Алгоритм S имеет временную сложность $O(nm\ln(m))$.

Доказательство. Оценим сложность формирования $S_{j_1 j_2 \ldots j_t}$ для каждой переменной x_i.

На шаге 1.1 выполняется $O(m)$ операций присваивания для формирования двумерного массива.

На шаге 1.2 — $O(m\ln(m))$ операций для сортировки массива.

На шаге 1.3 — $O(m)$ операций для прохода по массиву, формирования индекса и заполнения соответствующего коэффициента в $S_{j_1 j_2 \ldots j_t}$.

То есть составление $S_{j_1 j_2 \ldots j_t}$ требует не более $O(nm\ln(m))$ операций.

1.4. Вычисление значений полиномов с учётом совпадающих частей

Для вычисления j-го полинома нужно сложить результаты вычисления тех $S_{j_1 j_2 \ldots j_t}$, в номерах которых есть j.

Продолжение примера 1.

$$\begin{aligned} P_1(x_1,\ldots,x_7) &= S_1(x_1,\ldots,x_7) + S_{124}(x_1,\ldots,x_7) + S_{13}(x_1,\ldots,x_7),\\ P_2(x_1,\ldots,x_7) &= S_{124}(x_1,\ldots,x_7) + S_{234}(x_1,\ldots,x_7) + S_{24}(x_1,\ldots,x_7),\\ P_3(x_1,\ldots,x_7) &= S_{13}(x_1,\ldots,x_7) + S_{234}(x_1,\ldots,x_7),\\ P_4(x_1,\ldots,x_7) &= S_{124}(x_1,\ldots,x_7) + S_{234}(x_1,\ldots,x_7) + S_{24}(x_1,\ldots,x_7). \end{aligned}$$

Для вычисления S_1 требуется одно сложение и одно умножение. Для вычисления S_{124}, S_{13}, S_{234} — по одной операции умножения. Для вычисления S_{24} требуется два умножения и одно сложение.

Для вычисления с помощью суммирования $S_{j_1 j_2 \ldots j_t}$ значений полиномов P_1, P_2, P_3 и P_4 требуется 2, 2, 1 и 2 операции сложения соответственно.

Всего требуется 9 сложений и 6 умножений. В то время как без использования схемы с $S_{j_1 j_2 \ldots j_t}$ для вычисления значений полиномов P_1, P_2, P_3 и P_4 нужно 10 сложений и 13 умножений.

Важно отметить, что есть ещё затраты на собственно поиск совпадающих частей линейных полиномов. Но поиск совпадающих частей линейных полиномов — это разовое действие, а последующее использование порожденных полиномов это многократное действие, поэтому при многократном вычислении системы выполняется меньше операций.

В полиноме P_1 всего 4 ненулевых коэффициента: 1 при x_1, 2 при x_5, 3 при x_6 и 2 при x_7. В полиномах P_2, P_3 и P_4 соответственно 4, 2 и 4 ненулевых коэффициента.

Итак, в четырёх полиномах P_1, P_2, P_3 и P_4 в совокупности 14 ненулевых коэффициентов.

В полиномах S_1, S_{124}, S_{13}, S_{234}, S_{24} соответственно 2, 1, 1, 1, 2 ненулевых коэффициентов.

Итак, в пяти полиномах S_1, S_{124}, S_{13}, S_{234}, S_{24} в совокупности лишь 7 ненулевых коэффициентов.

2. Возможность использования линейных полиномов с целыми коэффициентами при классификации

2.1. Классификация и байесовский классификатор

Задача классификации — формализованная задача, в которой имеется множество объектов, разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Принадлежность к классам остальных объектов не известна. Требуется построить алгоритм, способный классифицировать, то есть указать номер (или наименование) класса, к которому относится произвольный объект из исходного множества.

Для классификации текстов можно применять байесовский классификатор (подробнее см., например, [**13**]). При одинаковой вероятности встретить текст любого класса работа этого классификатора сводится к вычислению произведений:

$$P(K_j|Text) = \prod_i P(w_i|K_j),$$

где $P(w_i|K_j)$ — вероятность встретить словосочетание w_i в текстах класса K_j. При этом вычисляется произведение вероятностей всех словосочетаний текста $Text$.

Словосочетания обычно используются, чтобы учесть корреляцию признаков. Употребляется термин *биграмма* — сочетание двух последовательно расположенных в тексте слов.

Вероятности $P(w_i|K_j)$ часто аппроксимируют как

$$P(w_i|K_j) = \frac{n_i + 1}{N_j},$$

где n_i — количество словосочетаний w_i во всех текстах класса K_j обучающей выборки, N_j — количество всех словосочетаний во всех текстах класса K_j обучающей выборки. Прибавление единицы осуществляется чтобы избежать появления словосочетаний с нулевыми вероятностями.

Рассматривают следующие задачи:

1. Выбор из нескольких альтернатив. В этом случае требуется отнести некоторый $Text$ ровно к одному из классов K_j, $j = 1, \ldots, n$. Считается n произведений $P(K_j|Text)$ и текст относится к тому классу, для которого вероятность $P(K_j|Text)$ получилась наибольшей.

2. Выбор из двух альтернатив: принадлежит или не принадлежит. В этом случае требуется отнести текст к классу или не отнести. Соответственно, считаются только два произведения.

2.2. Переход к линейным полиномам с целыми коэффициентами

При достаточно большой длине текста в формуле из предыдущего параграфа придется перемножать большое количество очень маленьких чисел. При этом, чтобы избежать появления машинного нуля пользуются свойством логарифма произведения:

$$\ln P(K_j|Text) = \sum_i \ln P(w_i|K_j).$$

Мы предлагаем округлять $\ln P(w_i|K_j)$ до целых чисел.

В ряде случаев это существенно не влияет на результат классификации. Например, есть задачи классификации текстов, содержащих много «мусорной информации», в таких задачах выделяют отдельную подзадачу — отбор только важных словосочетаний. Есть также задачи классификации сильно отличающихся друг от друга текстов.

В подобных случаях можно ожидать, что округления существенно не повлияют на результат.

Мы также предлагаем вычитать из всех округленных $\ln P(w_i|K_j)$ минимальное из них. Это преобразование без изменения свойств равенств позволяет складывать положительные, а не отрицательные числа.

Пример 2. Допустим, словосочетание «водительские права» (w_i) встретилось 63 раза среди всех 2460 словосочетаний класса «водитель» (K_j) обучающей выборки, то есть $n_i = 63$, $N_j = 2460$. Тогда

$$P(w_i|K_j) = \frac{n_i + 1}{N_j} = \frac{63 + 1}{2460} = 0.026.$$

Далее имеем $\ln P(w_i|K_j) = \ln 0.026 = -3.65$. Округляя это число, получаем -4.

Минимальное из значений логарифмов вероятностей

$$\ln \frac{1}{2460} = \ln 0.0004065 = -7.808.$$

Округляя это число, получаем -8.

Далее вычитаем $-4 - (-8) = 4$. Это окончательный коэффициент для словосочетания «водительские права», полученный на этапе обучения.

Итак, если при определении принадлежности произвольного текста не из обучающей выборки в этом тексте встретится словосочетание «водительские права», сумма для класса «водитель» будет увеличена на 4.

Замечание. Если решается задача выбора из двух альтернатив, то можно коэффициенты словосочетаний, которые говорят о принадлежности классу, считать положительными, а коэффициенты, которые говорят о непринадлежности классу, — отрицательными. Если сумма получилась больше нуля, сказать о принадлежности текста классу, а если меньше — о непринадлежности.

2.3. Об экономии памяти

1. В результате преобразований и округлений самые редко встречающиеся словосочетания будут иметь нулевые коэффициенты. Такие словосочетания и коэффициенты не нужно хранить. Количество обнуляемых коэффициентов зависит от конкретной задачи, особенностей конкретных текстов.

2. Вместо вещественного значения каждого коэффициента нужно хранить небольшое целое число.

3. После поиска совпадающих частей в полиномах количество коэффициентов уменьшается на

$$\sum_{1 \leqslant j_1 < j_2 < \ldots < j_t \leqslant n} (t-1) f(S_{j_1 j_2 \ldots j_t}),$$

где $f(S_{j_1 j_2 \ldots j_t})$ — количество ненулевых коэффициентов в $S_{j_1 j_2 \ldots j_t}$. При этом количество хранимых полиномов увеличивается на $S - m$, где S — количество ненулевых $S_{j_1 j_2 \ldots j_t}$. Если добавляется новый полином, то обязательно отпадает необходимость хранить хотя бы одно словосочетание, в результате общее количество хранимых словосочетаний и имен полиномов не растёт.

Выигрыш в памяти определяется сходством полиномов, которое в свою очередь определяется сходством понятий, соответствующих полиномам.

Пример 3. Рассмотрим преобразования, происходящие с частью словосочетаний и коэффициентов классов «водитель» и «швея», и возникающую в связи с этим экономию памяти.

вещественные	целые	итоговые
водитель	*водитель*	*водитель&швея*
график работ	график работ	график работ
-4,58949	3	3
рабочий день	рабочий день	рабочий день
-4,99495	3	3
петербург ленинградск		
-7,88532		
швея	*швея*	
график работ	график работ	
-4,59930	3	
рабочий день	рабочий день	
-4,74048	3	

Вначале формируются вещественные коэффициенты байесовского классификатора (левый столбец таблицы). В примере рассмотрены три словосочетания класса «водитель»: «график работ», «рабочий день», «петербург ленинградск», а также два словосочетания класса «швея»: «график работ», «рабочий день». Указаны получившиеся в

результате вычислений байесовского классификатора коэффициенты для этих словосочетаний.

1. Затем эти коэффициенты округляются. Например, коэффициент для словосочетания «график работ» в результате округлений становится равен -5, а коэффициент для словосочетания «петербург ленинградск» в результате округления становится равен -8. Из всех коэффициентов вычитается минимальное округлённое значение логарифмов вероятностей, в примере -8. Так получается второй столбец таблицы. В нём нет получившей нулевой коэффициент фразы «петербург ленинградск».

2. Коэффициенты второго столбца представляют собой небольшие целые числа (в данном примере тройки).

3. Далее осуществляется поиск совпадающих коэффициентов. В последнем столбце таблицы приведёна уже совпадающая часть для водителя и швеи. То есть вместо четырёх словосочетаний нужно хранить два.

3. Вычислительные эксперименты

Долю совпадающих коэффициентов системы полиномов **определим** как отношение количества коэффициентов a_{ij} системы из параграфа 1.1, вошедших в какую-то совпадающую часть $S_{j_1\ldots j_t}$, $t \geqslant 2$, и количества всех ненулевых коэффициентов этой системы.

Эксперимент 1. Для обучения были взяты шесть классов с вакансиями: швея, уборщица, грузчик, упаковщик, кассир, водитель. В каждом классе примерно по 60 вакансий.

В контрольной выборке были вакансии тех компаний, которых не было в обучающей.

В начале мы сравнили точность работы обычного байесовского классификатора и байесовского классификатора, коэффициенты которого округлялись с помощью описанного в этой статье способа. Точность вычислений (accuracy — доля правильных решений) обычного байесовского классификатора составила 73%. Точность вычислений классификатора с округлениями составила 77%. Из этого следует, что при использовании указанных классификаторов мы получили почти идентичные результаты. Размер базы данных с полиномами для обычного байесовского классификатора составил 719 Кбайт, а для классификатора с округлениями 79 Кбайт, доля совпадающих коэффициентов системы полиномов составила 26%.

Эксперимент 2. Стоит отметить, что тексты с объявлениями о вакансиях замусорены. Например, «метро вднх» не очень хорошо связано с тем, что нужен будет именно грузчик. Поэтому можно осуществить некоторую подстройку алгоритма, использованного на тех же данных, что и в первом эксперименте. А именно, сделать равными наименьшим вероятностям вероятности редких словосочетаний и вероятности словосочетаний, которые встречались в каждом типе вакансий.

В этом случае точность обоих методов составила 89%. Размер базы данных с полиномами для обычного байесовского классификатора составил 47 Кбайт, а для классификатора с округлениями 27 Кбайт, доля совпадающих коэффициентов системы полиномов составила 26%.

Для данной предметной области удаление малозначимых словосочетаний часто применяется, например, хорошо зарекомендовали себя применяемые в том числе и для этой цели методы, основанные на лексико-семантическом анализе и сингулярном разложении терм-документных матриц [**14**].

В **третьем эксперименте** проверялось распознавание очень длинных текстов, авторство для писателей разных лет. После обучения на текстах «Война и мир» Л.Н. Толстого и «Властелин колец» Д.Р.Р. Толкиена классификатор корректно отделял друг от друга произведения этих авторов. Размер базы данных с полиномами для обычного байесовского классификатора составил 5807 Кбайт, а для классификатора с округлениями 338 Кбайт, доля совпадающих коэффициентов системы полиномов составила 8%.

Заключение

Изучаемый в данной работе метод экономии памяти в задачах классификации текстов поиском совпадающих частей линейных полиномов вполне совместим с различными известными методами обработки текстов: алгоритмом быстрого поиска Ахо-Карасик [**15**], стеммингом [**16**] и распараллеливанием.

Итак, мы исследовали эвристический метод экономии памяти при классификации текстов. В результате наших экспериментов была продемонстрирована эффективность метода, то есть размер используемой базы данных сокращался при неизменном качестве классификации.

Список литературы

[1] Э. Г. Белага. «О вычислении значений многочлена от одного переменного с предварительной обработкой коэффициентов», Проблемы кибернетики, т. **5**, Физматгиз, М., 1961, с. 7–15. ↑ 133

[2] В. Я. Пан. «Некоторые схемы для вычисления значений полиномов с вещественными коэффициентами», Проблемы кибернетики, т. **5**, Физматгиз, М., 1961, с. 17–29. ↑ 133

[3] O. Bachmann, P. S. Wang, E. V. Zima. "Chains of Recurrences — a Method to Expedite the Evaluation of Closed-Form Functions", *Proc. of the ISSAC'94* (Oxford, United Kingdom, July 20–22, 1994), ACM Press, Oxford, UK, July 1994, pp. 242–249. ↑ 133

[4] С. Н. Селезнева. «Сложность систем функций алгебры логики и систем функций трехзначной логики в классах поляризованных полиномиальных форм», *Дискретная математика*, **27**:1 (2015), с. 111–122. ↑ 133

[5] Н. К. Маркелов. «Нижняя оценка сложности функций трехзначной логики в классе поляризованных полиномов», *Вестник Московского университета. Серия 15. Вычислительная м математика и кибернетика*, 2012, №3, с. 40–45. ↑ 133

[6] Т. М. Косовская. «Самообучающаяся сеть с ячейками, реализующими предикатные формулы», *Труды СПИИРАН*, 2015, №6(43), с. 94–113.↑ 133

[7] Н. Ф. Алексиадис. «Алгоритмическая неразрешимость проблемы полноты для полиномов с целыми коэффициентами», *Вестник МЭИ*, 2015, №3, с. 110–117. ↑ 133,134

[8] А. И. Мамонтов, Д. Г. Мещанинов. «Проблема полноты в функциональной системе линейных полиномов с целыми коэффициентами», *Дискретная математика*, **22**:4 (2010), с. 64–82. ↑ 133

[9] А. И. Мамонтов, Д. Г. Мещанинов. «Алгоритм распознавания полноты в функциональной системе $L(\mathbb{Z})$», *Дискретная математика*, **26**:1 (2014), с. 85–95. ↑ 133

[10] A. N. Gorban, D. C. Wunsch II. "The General Approximation Theorem", *Proceedings IJCNN'98* (Anchorage, Alaska, USA, May 4–9, 1998), IEEE, Oxford, UK, 1998, pp. 1271–1274. ↑ 133

[11] В. С. Половников. «О нелинейных характеристиках нейронных схем в произвольных базисах», *Интеллектуальные системы*, **17**:1-4 (2013), с. 87–90. ↑ 133

[12] А. И. Мамонтов. «Об использующем суперпозиции способе эффективного вычисления систем линейных полиномов с целыми коэффициентами», *Интеллектуальные системы. Теория и приложения*, **20**:3 (2016), с. 58–63. ↑ 134

[13] Т. Сегаран. *Программируем коллективный разум*, Символ-Плюс, СПб, 2008, 368 с. ↑ 138

[14] Д. В. Бондарчук. *Алгоритмы интеллектуального поиска на основе метода категориальных векторов*, Дис. ... к.ф.-м.н., Уральский государственный университет путей сообщения, Екатеринбург, 2016, 141 с. ↑ 143

[15] A. V. Aho, M. J. Corasick. "Efficient string matching: An aid to bibliographic search", *Communications of the ACM*, **18**:6 (1975), pp. 333–340. ↑ 143

[16] M. F. Porter. "An algorithm for suffix stripping", *Program*, **14**:3 (1980), pp. 130–137. ↑ 143

Рекомендовал к публикации *д.ф.-м.н. С. В. Знаменский*

Пример ссылки на эту публикацию:

А. И. Мамонтов, С. М. Рябинов. «Об одном методе экономии памяти при классификации текстов», *Программные системы: теория и приложения*, 2017, **8**:4(35), с. 133–147.

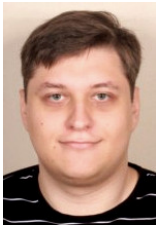
URL: http://psta.psiras.ru/read/psta2017_4_133-147.pdf

Об авторах:

Андрей Игоревич Мамонтов

к.т.н., доцент кафедры математического моделирования НИУ «МЭИ». Основные научные интересы: функциональные системы линейных полиномов, моделирование логических вычислений, построение нейронных сетей, автоматическая классификация текстов.

e-mail: mamontovai@mpei.ru



Станислав Михайлович Рябинов

выпускник кафедры математического моделирования НИУ «МЭИ». Основные научные интересы: автоматическая классификация текстов, нейронные сети.

e-mail: insiderxxl@mail.ru

Andrey Mamontov, Stanislav Rjabinov. *On one method of saving memory when classifying texts.*

Abstract. The article investigates the method of memory saving in tasks of classification of texts by searching for matching parts of linear polynomials. The algorithm for finding matching parts in linear polynomials with integer coefficients is given at the beginning. This algorithm makes it possible to calculate systems of linear polynomials with integer coefficients more quickly and use less memory for their storage. The algorithm is then used to find the matching parts of the linear polynomials that arise when classifying texts using the Bayesian classifier. We provide computational experiments that show memory saving. (*In Russian*).

Key words and phrases: text classification, linear polynomials, integers, Bayes classifier.

References

[1] E. G. Belaga. "Evaluation of polynomials of one variable with preliminary processing of the coefficients", Problems of Cybernetics, vol. **5**, Pergamon Press, 1961, pp. 1–13.

[2] V. Ya. Pan. "Some schemes for the evaluation of polynomials with real coefficients", Problems of Cybernetics, vol. **5**, Pergamon Press, 1961, pp. 14–32.

[3] O. Bachmann, P. S. Wang, E. V. Zima. "Chains of Recurrences — a Method to Expedite the Evaluation of Closed-Form Functions", *Proc. of the ISSAC'94* (Oxford, United Kingdom, July 20–22, 1994), ACM Press, Oxford, UK, July 1994, pp. 242–249.

[4] S. N. Selezneva. "Complexity of systems of functions of Boolean algebra and systems of functions of three-valued logic in classes of polarized polynomial forms", *Discrete Math. Appl.*, **26**:2 (2016), pp. 115–124.

[5] N. K. Markelov. "A lower estimate of the complexity of three-valued logic functions in the class of polarized polynomials", *Moscow University Computational Mathematics and Cybernetics*, **36**:3 (2012), pp. 150–154.

[6] T. M. Kosovskaya. "Self-training Network with the Sells Implementing Predicate Formulas", *Trudy SPIIRAN*, 2015, no.6(43), pp. 94–113 (in Russian).

[7] N. F. Aleksiadis. "Algorithmic Unsolvability of Completeness Problem for Polynomials with Integer Coefficients", *Vestnik MEI*, 2015, no.3, pp. 110–117 (in Russian).

[8] A. I. Mamontov, D. G. Meshchaninov. "The Completeness Problem in the Function Algebra of Linear Integer-Coefficient Polynomials", *Discrete Math. Appl.*, **20**:5–6 (2010), pp. 621–641.

[9] A. I. Mamontov, D. G. Meshchaninov. "The algorithm for completeness recognizing in function algebra $L(\mathbb{Z})$", *Discrete Math. Appl.*, **24**:1 (2014), pp. 21–28.

[10] A. N. Gorban, D. C. Wunsch II. "The General Approximation Theorem", *Proceedings IJCNN'98* (Anchorage, Alaska, USA, May 4–9, 1998), IEEE, Oxford, UK, 1998, pp. 1271–1274.

[11] V. S. Polovnikov. "On nonlinear characteristics of neural circuits in arbitrary bases", *Intellektual'nyye sistemy*, **17**:1-4 (2013), pp. 87–90 (in Russian).

DOI: 10.25209/2079-3316-2017-8-4-133-147

[12] A. I. Mamontov. “On the method of efficient calculation of systems of linear polynomials with integer coefficients that use superpositions”, *Intellektual'nyye sistemy. Teoriya i prilozheniya*, **20**:3 (2016), pp. 58–63 (in Russian).

[13] T. Segaran. *Programming Collective Intelligence*, O'Reilly Media, 2008, 368 p.

[14] D. V. Bondarchuk. *Intelligent search algorithms based on the method of categorical vectors*, Dis. . . . k.f.-m.n., Ural'skiy gosudarstvennyy universitet putey soobshcheniya, Yekaterinburg, 2016 (In Russian), 141 p.

[15] A. V. Aho, M. J. Corasick. “Efficient string matching: An aid to bibliographic search”, *Communications of the ACM*, **18**:6 (1975), pp. 333–340.

[16] M. F. Porter. “An algorithm for suffix stripping”, *Program*, **14**:3 (1980), pp. 130–137.

Sample citation of this publication:

Andrey Mamontov, Stanislav Rjabinov. “On one method of saving memory when classifying texts”, *Program systems: Theory and applications*, 2017, **8**:4(35), pp. 133–147. (*In Russian*).

URL: http://psta.psiras.ru/read/psta2017_4_133-147.pdf