УДК 004.021
DOI: 10.24160/1993-6982-2020-4-129-135

О способах экономии памяти при решении задач классификации линейным методом опорных векторов

А.И. Мамонтов

Рассмотрена автоматическая классификация с использованием линейного метода опорных векторов с целыми коэффициентами, получаемыми при помощи округления вещественных. С помощью экспериментальных расчетов на реальном примере продемонстрировано, как можно достичь сокращения требуемого объёма памяти вычислительной системы за счёт представления использующихся в классификаторе целочисленных линейных полиномов.
Изучены и применены два способа: способ, основанный на экономном представлении совпадающих частей линейных полиномов, и способ, базирующийся на экономном представлении наиболее часто встречающихся в полиномах коэффициентов.
Для экспериментальных расчетов взяты наборы данных Reuters-21578 и 20Newsgroups. Отмечено, что в эксперименте с набором Reuters-21578 количество необходимых для классификации признаков оптимизировалось, а в эксперименте с набором 20Newsgroups — нет.
В эксперименте с набором данных Reuters-21578 для представления коэффициентов полиномов использовано восемь бит. При хранении коэффициентов без округлений необходимо 3,81 Мб. Хранение округленных коэффициентов без применения предложенных способов требует 0,48 Мб, с использованием первого способа — 0,16 Мб, а второго — 0,148 Мб. В сжатом виде хранение округленных коэффициентов требует 0,12 Мб при первом способе и 0,09 Мб — при втором.
В эксперименте с набором 20Newsgroups для представления коэффициентов полиномов использовали двенадцать бит. При хранении коэффициентов без округлений нужно 19,86 Мб. При хранении округленных коэффициентов без применения способов — 4,96 Мб, при использовании первого способа — 4,2 Мб, второго — 3,98 Мб. При хранении округленных коэффициентов в сжатом виде первым способом потребовалось 1,45 Мб, а вторым — 1,27 Мб.
Использование результатов работы возможно при разработке программно-аппаратных средств для автоматической классификации.

Ключевые слова: линейные полиномы, целые числа, метод опорных векторов, классификация.

Для цитирования: Мамонтов А.И. О способах экономии памяти при решении задач классификации линейным методом опорных векторов // Вестник МЭИ. 2020. № 4. С. 129—135. DOI: 10.24160/1993-6982-2020-4-129-135.

On Memory Saving Methods in Solving Classification Problems by Using the Linear Support Vector Machine Techniques

A.I. Mamontov

Automatic classification carried out using the linear support vector machine techniques with integer coefficients obtained from real ones by rounding them is considered. It is demonstrated, by means of experimental calculations for a real example, how the required amount of computing system memory can be reduced by representing the integer linear polynomials used in the classifier.
Two methods are considered: a method based on an economical representation of the matching parts of linear polynomials and a method based on an economical representation of the coefficients most frequently encountered in the polynomials.
For experimental computations, the Reuters-21578 and 20Newsgroups data sets were used. In the experiment with the Reuters-21578 data set, the number of features necessary for classification was optimized, and in the experiment with the 20Newsgroups data set, this was not done.
In the experiment with the Reuters-21578 data set, eight bits were used to represent polynomial coefficients. In storing coefficients without rounding, 3.81 MB of memory space is needed. For storing the rounded coefficients without using the proposed methods, 0.48 MB of memory space is required; in the cases of using the first and second methods, 0.16 and 0.148 MB of memory space, respectively, are required. For storing rounded coefficients in a compressed form, 0.12 and 0.09 MB of memory space, respectively, are required in the cases of using the first and second methods.
In the experiment with the 20Newsgroups data set, twelve bits were used to represent polynomial coefficients. For storing coefficients without rounding, 19.86 MB of memory space is required. For storing rounded coefficients without using the proposed methods, 4.96 MB of memory space is required; in the cases of using the first and second methods, 4.2 and 3.98 MB of memory space, respectively, are required. For storing rounded coefficients in a compressed form, 1.45 and 1.27 MB of memory space, respectively, are required in the cases of using the first and second methods.
The obtained study results can be used in developing software and hardware for solving automatic classification problems.

Key words: linear polynomials, integers, support vector machine, classification.

For citation: Mamontov A.I. On Memory Saving Methods in Solving Classification Problems by Using the Linear Support Vector Machine Techniques. Bulletin of MPEI. 2020;4:129—135. (in Russian). DOI: 10.24160/1993-6982-2020-4-129-135.

Введение

Рассмотрена автоматическая классификация с помощью линейного метода опорных векторов с целыми коэффициентами и продемонстрировано, как можно сократить требуемый объём памяти вычислительной системы за счёт представления использующихся в классификаторе полиномов.

Классификация с использованием метода опорных векторов выполняется на компьютерах с высокой числовой точностью, то есть с использованием вычислений с плавающей запятой двойной точности. Однако, в результате энергетических и вычислительных ограничений, маломощные и интегрированные приложения, реализованные на встроенных системах, требуют алгоритмов низкой сложности. В публикациях [1, 2] упоминаются примеры подобных приложений: классификация слуховых сцен в слуховых аппаратах, вычислениях в спутниках, сенсорные сети. Для эффективной работы этим приложениям необходим компромисс между точностью и сложностью алгоритма. Решением задачи, по мнению авторов [1, 2], является использование чисел с фиксированной точкой (которые можно представить с помощью целых чисел).

Ограничение памяти в приложениях, реализуемых на встроенных архитектурах, может стать серьёзным препятствием для достижения эффективности информационной системы. Оно возможно и при распознавании на обычных вычислительных устройствах. В качестве примера приведем программу распознавания новостных потоков, работающую в сети Интернет на виртуальном хостинге с малым объёмом памяти для базы данных.

Методы решения задач, связанных с ограничением памяти, встречаются в распознавании, среди них отметим применение кодов Хафмана [3] и материалы по разреженным матрицам [4].

При работе с целочисленными параметрами в линейном методе опорных векторов, по нашему мнению, интересна задача сокращения требуемого объёма памяти вычислительной системы с помощью рассмотрения разнообразных композиций (суперпозиций) полиномов с целыми коэффициентами.

Следует отметить, что линейный метод опорных векторов — современный и достаточно популярный метод распознавания. Работы по линейному методу опорных векторов достаточно распространены в научной среде. Научный интерес представляет [5], в которой рассматривается линеаризация для многих задач, то есть приведение задач к условиям, в которых можно использовать линейный метод опорных векторов.

Рассмотрим основные труды, посвящённые композициям полиномов. В научной литературе по теме исследования достаточно распространены статьи, посвященные различным представлениям полиномов, выгодным с вычислительной точки зрения. Например, в [6, 7] проанализированы схемы вычисления вещественных полиномов, в [8, 9] — сложность полиномиальных представлений k-значных функций. Представление с помощью подформул полиномов с целыми коэффициентами представлено в [10 — 17]. Авторы [18 — 20] изучали представление с помощью линейных и нелинейных полиномов искусственных нейронных сетей — конструкций, применяемых при классификации.

Перспективные целочисленные модели распознавания, использующие полиномы, описаны в [1, 2, 21 — 24]. Исследование [1] посвящено методу опорных векторов, в котором некоторые параметры являются целочисленными, авторы [21] рассматривают метод опорных векторов, в котором нет умножений, а есть только сдвиги, сложения и вычитания. Этот метод адаптирован для использования в архитектурах с быстрыми операциями сложения, вычитания и сдвигов целых чисел. В работах [2, 22] предложен байесовский классификатор с целочисленными параметрами, а в [23, 24] — искусственные нейронные сети с целочисленными параметрами.

Настоящее исследование опирается на ранее разработанные для байесовского классификатора методы [16, 17]. Автор развил и дополнил их. Итак, рассматривается линейный метод опорных векторов, его параметры-коэффициенты линейных полиномов округляются и представляются целыми числами. На следующем этапе получившиеся линейные полиномы хранятся в памяти в виде композиций более простых полиномов. В результате получены представления полиномов с целыми коэффициентами, позволяющими использовать меньше коэффициентов, то есть сокращать требуемый объём памяти вычислительной системы. Объём памяти уменьшается за счёт снижения избыточности, присутствующей в промежуточных данных.

О линейном методе опорных векторов

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. Оно называется обучающей выборкой. Принадлежность к классам остальных объектов неизвестна. Требуется на основе обучающей выборки построить алгоритм, способный классифицировать, т. е. указать номер (или наименование) класса, к которому относится произвольный объект из исходного множества. Любой объект может относиться сразу к нескольким классам.

Качество работы данного алгоритма можно оценивать с помощью различных мер. В настоящей работе качество работы алгоритма оценивается с помощью меры Accuracy — процента правильно классифицированных объектов из контрольной выборки.

Пусть объекты описываются $(n + 1)$-компонентными векторами $\mathbf{x} = (x_1, x_2, \ldots, x_n, 1) \in \mathbf{R}^n$ n-мерного пространства (n+1-я компонента добавлена для удобства вычислений). Линейная решающая функция в общем случае представляется следующим образом [25]:

$$d(\mathbf{x})=(\mathbf{a}, \mathbf{x}) = a_1x_1 + a_2x_2 + ... + a_nx_n + a_{n+1},$$

где $\mathbf{a} = (a_1, a_2, \ldots, a_n, a_{n+1})$ — весовой вектор.

При делении на два класса линейная решающая функция определяет решающие правила:

— если $d(\mathbf{x}) > 0$, то объект **x** относится к первому классу;

— если $d(\mathbf{x}) \le 0$, то объект **x** относится ко второму классу.

Рассмотрим применение классификации линейного метода опорных векторов, в котором линейные решающие функции строятся с помощью библиотеки liblinear [26].

Рассматриваемые объекты могут принадлежать любому подмножеству множества классов $K_1, K_2, \ldots, K_q$, поэтому для каждого класса строится своя линейная решающая функция:

$$d_i(\mathbf{x}) = (\mathbf{a}_i, \mathbf{x}) = a_{i1}x_1 + a_{i2}x_2 + ... + a_{i,n}x_n + a_{i,n+1}, i = 1, 2, \ldots, q.$$

Коэффициенты a_{ij} округляли и представляли целыми числами (числами с фиксированной точкой). Подобный приём применялся в [1, 2, 22]. Чтобы не вводить новые обозначения обозначим округленные коэффициенты a_{ij} (в настоящей работе, если не указано иное, использованы только линейные функции с округленными коэффициентами a_{ij}).

Задача представления и хранения линейных решающих функций

На следующем этапе экспериментального расчета встает задача экономного представления в памяти линейных полиномов — решающих функций $d_i(\mathbf{x})$, после того, как эти решающие функции были построены. Осуществим представление за счёт композиций (суперпозиций) функций, а именно:

— выполним поиск некоторых простых частей в решающих функциях;

— представим решающие функции в виде композиций простых частей.

Следует отметить, что подобные представления сохраняют возможность вычисления функций.

Следовательно, необходимо разработать способы хранения решающих функций, которые:

— будут экономны по памяти;

— по возможности, позволят вычислять решающие функции за не очень большое время.

С математической точки зрения подобные способы хранения будут композициями (суперпозициями) линейных полиномов.

Способы хранения решающих функций разработаны в условиях влияния следующих ограничений.

Многие коэффициенты a_{ij} совпадают в нескольких линейных решающих функциях при одинаковых номерах j. Например, такая ситуация получается, когда ряд признаков оказывает похожее влияние на принадлежность или непринадлежность объекта сразу нескольким классам.

Некоторые коэффициенты в линейных решающих функциях равны одному и тому же числу. Подобное случается, когда некоторые признаки имеют малое влияние на принадлежность объекта классу.

Рассмотрим случай наличия большого количества нулевых коэффициентов среди a_{ij}, т. е. наличия признаков, влияющих на принадлежность объекта одним классам, и не оказывающих влияния на принадлежность объекта другим классам.

Способы хранения решающих функций

Первый способ хранения решающих функций. Построим новые линейные полиномы $e_{i1\ldots it}(x)$, в которые входят те, и только те мономы, которые входят также в каждую из решающих функций с номерами $i_1, \ldots, i_t$ и отсутствуют в остальных решающих функциях.

Данные полиномы можно построить с помощью алгоритма полиномиальной сложности [16].

Хранение линейных полиномов может быть организовано так, что невыгодно создавать линейные полиномы $e_{i1i2}(x)$, построенные на основе совпавших частей двух решающих функций. В этом случае их можно удалить, а входящие в них мономы поместить в полиномы $e_{i1}(x)$, $e_{i2}(x)$.

Линейный полином при небольшом количестве ненулевых коэффициентов сохраним в виде списков пар $<a_{ij}, j>$ (ненулевой коэффициент, номер этого коэффициента).

При большом количестве ненулевых коэффициентов линейный полином выгоднее хранить в виде пары списков:

— список ненулевых коэффициентов a_{ij};

— список из нулей и единиц, j-й элемент которого равен 1, если соответствующий коэффициент $a_{ij} \ne 0$, если же $a_{ij} = 0$, то j-й элемент равен 0.

Для каждого образованного линейного полинома $e_{i1\ldots it}(x)$ нужно хранить также список номеров решающих функций, мономы из которых вошли в его состав.

Пример 1. Для линейных решающих функций

$$d_1(\mathbf{x}) = 2x_1 + 3x_2 + 5x_3;\ d_2(\mathbf{x}) = 1x_1 + 3x_2 + 3x_3$$

строим новые линейные полиномы:

$$e_1(\mathbf{x}) = 2x_1 + 5x_3;\ e_2(\mathbf{x}) = 1x_1 + 3x_3;\ e_{12}(\mathbf{x}) = 3x_2.$$

Списки пар $<a_{ij}, j>$ для этих линейных полиномов:

$e_1(\mathbf{x})$: <2,1>, <5,3>; $e_2(\mathbf{x})$: <1,1>, <3,3>; $e_{12}(\mathbf{x})$:<3,2>.

Пары списков для данных полиномов:

$e_1(\mathbf{x})$: 1. (2, 5), 2. (1, 0, 1); $e_2(\mathbf{x})$: 1. (1, 3), 2. (1, 0, 1);
$e_{12}(\mathbf{x})$: 1. (3), 2. (0, 1, 0).

Второй способ хранения решающих функций. Упорядочим все значения коэффициентов a_{ij} во всех решающих функциях d_i по убыванию количества присутствий этих значений коэффициентов в функциях d_i, получим список чисел $b_1, b_2, \ldots, b_l$. Рассмотрим часть этого списка $b_1, b_2, \ldots, b_m$.

Для каждой решающей функции d_i построим $m + 1$-линейный полином:

— m полиномов, все коэффициенты a_{ij} каждого из которых равны b_k, $k = 1,\ldots, m$;

— полином, содержащий все коэффициенты исходного полинома, не равные ни одному из чисел $b_1, \ldots, b_m$.

Первые полиномы можно хранить в виде списка номеров коэффициентов j, а вторые —в виде списков пар $<a_{ij}, j>$ (ненулевой коэффициент, номер этого коэффициента).

При большом количестве коэффициентов, равных $b_1, \ldots, b_m$, в решающих функциях для каждой решающей функции d_i выгодно составить список из нулей и единиц, j-й элемент которого равен 1, если соответствующий коэффициент $a_{ij} = b_1$. Если же $a_{ij} \neq b_1$ (j-й элемент этого списка равен 0) — удалить из всех решающих функций коэффициенты, равные b_1 и получить новые линейные полиномы, коэффициенты которых не равны b_1, затем проделать ту же самую операцию с числами $b_2, \ldots, b_m$.

Пример 2. В линейных решающих функциях

$$d_1(\mathbf{x}) = 2x_1 + 3x_2 + 5x_3;\ \ d_2(\mathbf{x}) = 1x_1 + 3x_2 + 3x_3$$

коэффициент 3 встречается три раза, коэффициенты 1, 2, 5 — по одному разу. Получим список чисел $b_1, b_2, \ldots, b_l$: 3, 1, 2, 5. Рассмотрим часть этого списка $b_1, b_2, \ldots, b_m$: 3.

Для d_1 построим линейные полиномы $2x_1 + 5x_3$ и $3x_2$, а для d_2 — $1x_1$ и $3x_2 + 3x_3$. Их можно хранить в виде списков для d_1: список номеров коэффициентов, равных 3: (2); для оставшихся коэффициентов список пар (<2, 1>, <5, 3>); для d_2: список номеров коэффициентов, равных 3: (2, 3); для оставшихся коэффициентов список пар (<1, 1>).

Список из нулей и единиц для хранения информации о коэффициентах, равных 3, в функции d_1: (0, 1, 0); в функции d_2: (0, 1, 1).

Новые линейные полиномы, коэффициенты в которых не равны 3:

$$\widetilde{d}_1(\mathbf{x}) = 2x_1 + 5x_3;\ \widetilde{d}_2(\mathbf{x}) = 1x_1.$$

Замечание. Проанализируем хранение информации не для каждой решающей функции d_i, а для каждой компоненты x_j. Приведённые способы хранения функций модифицируются следующим образом:

— вместо пары $<a_{ij}, j>$ (ненулевой коэффициент, номер этого коэффициента) взята пара $<a_{ij}, ip>$ (ненулевой коэффициент, номер линейного полинома), в который входит данный коэффициент;

— не для каждого полинома, а для каждой компоненты x_j строятся списки пар $<a_{ij}, ip>$, номеров ip и списки нулей и единиц.

Результаты экспериментов

Для экспериментов использовали наборы данных Reuters-21578 и 20Newsgroups.

Набор данных Reuters-21578 — выборка новостных текстов. В нем каждый текст может принадлежать любому множеству из $q = 135$ классов. Для обучения использовали 7264 текстов, а для распознавания — 3113.

При применении 8-битовых чисел для представления округлённых коэффициентов a_{ij} при распознавании значение метрики Accuracy на 3113 текстах, требуемых для распознавания, оказалось не меньше, чем значение метрики Accuracy на этих же текстах, полученное при использовании чисел с двойной точностью.

Количество характеристик, используемых для представления классифицируемых объектов — текстов, оптимизировалось и составило $n = 3695$, поэтому для хранения изначальных решающих функций d_i с неокругленными коэффициентами потребовалось 3,81 Мб. Хранение округленных коэффициентов без применения наших способов заняло 0,48 Мб.

При первом способе хранения решающих функций нужно 0,12 Мб при хранении информации об a_{ij} в виде 8-битовых чисел на диске и несколько больше, из-за особенностей реализации, — 0,16 Мб при хранении тех же коэффициентов в оперативной памяти.

Исследована скорость вычисления построенных линейных полиномов $e_{i1\ldots it}$. Время вычисления значений всех решающих функций в 7 раз меньше времени, требуемого для расчета изначальных линейных решающих функций с помощью программы

```
for (j = 0; j < classiferscount; j++)
{
    for (k = 0; k < Ntest; k++)
    {
        d = 0;
        for (i = 0; i < featurescount; i++)
        {
            d += Xtest[k][i] * a[i][j];
        }
        d += a[featurescount][j];
        if (d > 0) r = 1;
        else r = –1;
        if (Ytest[k][j] == r)
        col = col + 1;
    }
}
```

При втором способе хранения наиболее часто в целочисленном представлении округленных решающих функций встречались следующие коэффициенты: 0, –1, 1, –2, 2, … В таблице 1 представлены результаты эксперимента.

Время вычисления значений оказалось в 10 раз меньше времени, требуемого для определения изначальных решающих функций.

Отметим, что сравнительно небольшое время, понадобившееся для вычисления решающих функций, доказало возможность получения решающих функций при таком представлении данных.

Существуют различные библиотеки, позволяющие быстро проводить вычисления линейной алгебры, например Eigen, а также способы представления исходных данных, помогающие увеличить скорость вычислений, но в данном примере нельзя исключить, что для подобных задач быстрые способы вычисления могут потребовать увеличения памяти.

Таблица 1

Результаты эксперимента с Reuters-21578

Коэффициенты, информация о которых хранится отдельно	Память для хранения информации о коэффициентах a_{ij} с использованием списков из нулей и единиц	Память для хранения полиномов в виде списков номеров коэффициентов j и пар $<a_{ij}, j>$
0	0,130	0,213
0, –1	0,113	0,188
0, –1, 1	0,103	0,172
0, –1, 1, –2	0,099	0,164
0, –1, 1, –2, 2	0,096	0,158
0, –1, 1, –2, 2, –3	0,094	0,155
0, –1, 1, –2, 2, –3, 3	0,093	0,152
0, –1, 1, –2, 2, –3, 3, –4	0,092	0,151
0, –1, 1, –2, 2, –3, 3, –4, 4	0,091	0,150
0, –1, 1, –2, 2, –3, 3, –4, 4, –5	0,091	0,149
0, –1, 1, –2, 2, –3, 3, –4, 4, –5, 5	0,091	0,148
0, –1, 1, –2, 2, –3, 3, –4, 4, –5, 5, –6	0,091	0,148
0, –1, 1, –2, 2, –3, 3, –4, 4, –5, 5, –6, 6	0,090	0,148

Набор данных 20Newsgroups — выборка новостных текстов. В нем каждый текст отнесен к одному из $q = 20$ классов. Для обучения использовано 11 314 текстов, а для распознавания — 7532.

При использовании 12-битовых чисел для представления округлённых коэффициентов a_{ij} при распознавании значение метрики Accuracy на 7532 текстах, взятых для анализа, оказалось не меньше, чем ее же значение на этих же текстах, полученное при использовании чисел с двойной точностью.

Количество характеристик, необходимых для представления классифицируемых объектов–текстов, не оптимизировано и составило n =1 30107, поэтому для хранения изначальных решающих функций d_i с неокругленными коэффициентами потребовалось 19,86 Мб. При хранении округленных коэффициентов без применения предложенных способов понадобилось 4,96 Мб.

При первом способе хранения решающих функций нужно 1,5 Мб для хранении информации об a_{ij} на диске в виде 12-битовых чисел и 3,72 Мб при хранении a_{ij} в виде чисел двухбайтового типа short в оперативной памяти.

При удалении полиномов $e_{i1i2}(x)$, построенных на основе совпадающих частей только двух функций, потребовалось 1,45 Мб при хранении информации об a_{ij} на диске в виде 12-битовых чисел и 4,2 Мб при хранении a_{ij} в виде чисел типа short в оперативной памяти.

При втором способе наиболее часто в целочисленном представлении округленных решающих функциях встречались следующие коэффициенты: 0, –1, –2, … . В таблице 2 показаны итоги эксперимента. Результаты о времени вычисления аналогичны предыдущему набору данных.

Следует отметить встречающийся на практике случай, когда при эксплуатации системы ожидаются данные с очень малым количеством ненулевых компонент x_j. В этом случае для быстрого получения информации для каждой компоненты указанные разнообразные списки следует создавать не для каждой решающей функции, а для каждой компоненты, то есть не по j, а по i. При этом используется аналогичный объем памяти при втором способе хранения информации обо всех коэффициентах a_{ij} с использованием списков из нулей и единиц. Поскольку предполагается небольшое количество вычислений, рекомендуем использовать именно этот способ в самом сжатом виде.

Заключение

Продемонстрированы способы, как при сохранении той же функциональности достичь сокращения требуемого объёма памяти вычислительной системы при классификации с помощью линейного метода опорных векторов с целыми коэффициентами.

Таблица 2

Результаты эксперимента с 20Newsgroups

Коэффициенты, информация о которых хранится отдельно	Память для хранения информации о коэффициентах a_{ij} с использованием списков из нулей и единиц	Память для хранения полиномов в виде списков номеров коэффициентов j и пар $<a_{ij}, j>$
0	1,47	4,65
0, –1	1,36	4,37
0, –1, –2	1,31	4,2
0, –1, –2, –3	1,28	4,07
0, –1, –2, –3, –4	1,27	3,98

Объём памяти сокращается за счёт уменьшения избыточности, присутствующей в промежуточных данных.

Результаты работы можно использовать при разработке программно-аппаратных средств для автоматической классификации.

Литература

1. **Anguita, D., Ghio, A., Pischiutta, S., Ridella, S.** A Support Vector Machine with Integer Parameters // Neurocomputing. 2008. V. 72. No. 1—3. Pp. 480—489.

2. **Tschiatschek S., Paul K., Pernkopf F.** Integer Bayesian Network Classifiers // Machine Learning and Knowledge Discovery in Databases. Lecture Notes in Computer Science. 2014. V. 8726. Pp. 209—224.

3. **Han S., Mao H., Dally W.J.** Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding // Proc. Intern. Conf. Learning Representation. San Juan, 2016. Pp. 1—14.

4. **Liu B., Wang M., Foroosh H., Tappen M., Penksy M.** Sparse Convolutional Neural Networks // IEEE Trans. Conf. Computer Vision and Pattern Recognition. Boston, 2015. Pp. 806—814.

5. **Lan L., Wang Z., Zhe S., Cheng W., Wang J., Zhang K.** Scaling Up Kernel SVM on Limited Resources: a Low-rank Linearization Approach // IEEE Trans. Neural Networks and Learning Systems. 2019. V. 30. Iss. 2. Pp. 369—378.

6. **Белага Э.Г.** О вычислении значений многочлена от одного переменного с предварительной обработкой коэффициентов // Проблемы кибернетики. 1961. Вып. 5. С. 7—15.

7. **Пан В.Я.** Некоторые схемы для вычисления значений полиномов с вещественными коэффициентами // Там же. С. 17—29.

8. **Селезнева С.Н.** Сложность систем функций алгебры логики и систем функций трехзначной логики в классах поляризованных полиномиальных форм // Дискретная математика. 2015. Т. 27. Вып. 1. С. 111—122.

9. **Маркелов Н.К.** Нижняя оценка сложности функций трехзначной логики в классе поляризованных полиномов // Вестник Московского университета. Серия «Вычислительная математика и кибернетика». 2012. Вып. 3. С. 40—45.

10. **Алексиадис Н.Ф.** Алгоритмическая неразрешимость проблемы полноты для полиномов с целыми коэффициентами // Вестник МЭИ. 2015. № 3. С. 110—117.

11. **Алексиадис Н.Ф.** Алгоритмическая неразрешимость задачи о нахождении базиса конечной полной системы полиномов с целыми коэффициентами // Интеллектуальные системы. Теория и приложения. 2016. Т. 20. Вып. 3. С. 19—23.

12. **Мамонтов А.И.** Применение функциональных систем полиномов при классификации и поиске информации // Вестник МЭИ. 2012. № 6. С. 117—123.

13. **Мамонтов А.И., Мещанинов Д.Г.** Проблема полноты в функциональной системе линейных полиномов с целыми коэффициентами // Дискретная математика. 2010. Т. 22. № 4. С. 64—82.

14. **Мамонтов А.И., Мещанинов Д.Г.** Алгоритм распознавания полноты в функциональной системе $L(Z)$ // Дискретная математика. 2014. Т. 26. № 1. С. 85—95.

References

1. **Anguita, D., Ghio, A., Pischiutta, S., Ridella, S.** A Support Vector Machine with Integer Parameters. Neurocomputing. 2008;72;1—3:480—489.

2. **Tschiatschek S., Paul K., Pernkopf F.** Integer Bayesian Network Classifiers. Machine Learning and Knowledge Discovery in Databases. Lecture Notes in Computer Science. 2014;8726:209—224.

3. **Han S., Mao H., Dally W.J.** Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. Proc. Intern. Conf. Learning Representation. San Juan, 2016:1—14.

4. **Liu B., Wang M., Foroosh H., Tappen M., Penksy M.** Sparse Convolutional Neural Networks. IEEE Trans. Conf. Computer Vision and Pattern Recognition. Boston, 2015:806—814.

5. **Lan L., Wang Z., Zhe S., Cheng W., Wang J., Zhang K.** Scaling Up Kernel SVM on Limited Resources: a Low-rank Linearization Approach. IEEE Trans. Neural Networks and Learning Systems. 2019;30;2:369—378.

6. **Belaga E.G.** O Vychislenii Znacheniy Mnogochlena ot Odnogo Peremennogo s Predvaritel'noy Obrabotkoy Koeffitsientov. Problemy Kibernetiki. 1961;5:7—15. (in Russian).

7. **Pan V.Ya.** Nekotorye Skhemy dlya Vychisleniya Znacheniy Polinomov s Veshchestvennymi Koeffitsientami. Tam zhe:17—29. (in Russian).

8. **Selezneva S.N.** Slozhnost' Sistem Funktsiy Algebry Logiki i Sistem Funktsiy Trekhznachnoy Logiki v Klassakh Polyarizovannykh Polinomial'nykh Form. Diskretnaya Matematika. 2015;27;1:111—122. (in Russian).

9. **Markelov N.K.** Nizhnyaya Otsenka Slozhnosti Funktsiy Trekhznachnoy Logiki v Klasse Polyarizovannykh Polinomov. Vestnik Moskovskogo Universiteta. Seriya «Vychislitel'naya Matematika i Kibernetika». 2012;3:40—45. (in Russian).

10. **Aleksiadis N.F.** Algoritmicheskaya Nerazreshimost' Problemy Polnoty dlya Polinomov s Tselymi Koeffitsientami. Vestnik MEI. 2015;3:110—117. (in Russian).

11. **Aleksiadis N.F.** Algoritmicheskaya Nerazreshimost' Zadachi o Nakhozhdenii Bazisa Konechnoy Polnoy Sistemy Polinomov s Tselymi Koeffitsientami. Intellektual'nye Sistemy. Teoriya i Prilozheniya. 2016;20;3:19—23. (in Russian).

12. **Mamontov A.I.** Primenenie Funktsional'nykh Sistem Polinomov pri Klassifikatsii i Poiske Informatsii. Vestnik MEI. 2012;6:117—123. (in Russian).

13. **Mamontov A.I., Meshchaninov D.G.** Problema Polnoty v Funktsional'noy Sisteme Lineynykh Polinomov s Tselymi Koeffitsientami. Diskretnaya Matematika. 2010; 22;4:64—82. (in Russian).

14. **Mamontov A.I., Meshchaninov D.G.** Algoritm Raspoznavaniya Polnoty v Funktsional'noy Sisteme $L(Z)$. Diskretnaya Matematika. 2014;26;1:85—95. (in Russian).

15. **Мамонтов А.И.** Об использующем суперпозиции способе эффективного вычисления систем линейных полиномов с целыми коэффициентами // Интеллектуальные системы. Теория и приложения. 2016. Т. 20. Вып. 3. С. 58—63.

16. **Мамонтов А.И., Рябинов С.М.** Об одном методе экономии памяти при классификации текстов // Программные системы: теория и приложения. 2017. Т. 8. № 4(35). С. 133—147.

17. **Мамонтов А.И.** О повышении эффективности вычислений при классификации изображений // Вестник МЭИ. 2019. № 5. С. 129—134.

18. **Gorban A.N., Wunsch D.C.** The General Approximation Theorem // IEEE Trans. World Congress on Computational Intelligence. Anchorage, 1998. Pp. 1271—1274.

19. **Горбань А.Н.** Обобщенная аппроксимационная теорема и вычислительные возможности нейронных сетей // Сибирский журнал вычислительной математики. 1998. Т. 1. № 1. С. 11—24.

20. **Половников В.С.** О нелинейных характеристиках нейронных схем в произвольных базисах // Интеллектуальные системы. 2013. Т. 17. № 1—4. С. 87—90.

21. **Anguita D., Pischiutta S., Ridella S., Sterpi D.** Feed–forward SVM without Multipliers // IEEE Trans. Neural Networks. 2006. V. 17. Pp. 1328—1331.

22. **Tschiatschek S., Pernkopf F.** On Bayesian Network Classifiers with Reduced Precision Parameters // IEEE Trans. Pattern Analysis and Machine Intelligence. 2015. V. 37(4). Pp. 774—785.

23. **Yi Y., Hangping Z., Bin Z.** A New Learning Algorithm for Neural Networks with Integer Weights and Quantized Non-linear Activation Functions // Artificial Intelligence in Theory and Practice. Boston: Springer, 2008. Pp. 427—431.

24. **Shuang Wu, Guoqi Li, Feng Chen, Luping Shi.** Training and Inference with Integers in Deep Neural Networks // Proc. VI Intern. Conf. Learning Representations. 2018. Pp. 1—14.

25. **Вапник В.Н., Червоненкис А.Я.** Теория распознавания образов. М.: Наука, 1974.

26. **Fan R.-E., Chang K.-W., Hsieh C.-J., Wang X.-R., Lin C.-J.** LIBLINEAR: a Library for Large Linear Classification // J. Machine Learning Research. 2008. V. 9. Pp. 1871—1874.

15. **Mamontov A.I.** Ob Ispol'zuyushchem Superpozitsii Sposobe Effektivnogo Vychisleniya Sistem Lineynykh Polinomov s Tselymi Koeffitsientami. Intellektual'nye Sistemy. Teoriya i Prilozheniya. 2016;20;3:58—63. (in Russian).

16. **Mamontov A.I., Ryabinov S.M.** Ob Odnom Metode Ekonomii Pamyati pri Klassifikatsii Tekstov. Programmnye Sistemy: Teoriya i Prilozheniya. 2017;8; 4(35):133—147. (in Russian).

17. **Mamontov A.I.** O Povyshenii Effektivnosti Vychisleniy pri Klassifikatsii Izobrazheniy. Vestnik MEI. 2019;5:129—134. (in Russian).

18. **Gorban A.N., Wunsch D.C.** The General Approximation Theorem. IEEE Trans. World Congress on Computational Intelligence. Anchorage, 1998:1271—1274.

19. **Gorban' A.N.** Obobshchennaya Approksimatsionnaya Teorema i Vychislitel'nye Vozmozhnosti Neyronnykh Setey. Sibirskiy Zhurnal Vychislitel'noy Matematiki. 1998; 1;1:11—24. (in Russian).

20. **Polovnikov V.S.** O Nelineynykh Kharakteristikakh Neyronnykh Skhem v Proizvol'nykh Bazisakh. Intellektual'nye Sistemy. 2013;17;1—4:87—90. (in Russian).

21. **Anguita D., Pischiutta S., Ridella S., Sterpi D.** Feed–forward SVM without Multipliers. IEEE Trans. Neural Networks. 2006;17:1328—1331.

22. **Tschiatschek S., Pernkopf F.** On Bayesian Network Classifiers with Reduced Precision Parameters. IEEE Trans. Pattern Analysis and Machine Intelligence. 2015; 37(4):774—785.

23. **Yi Y., Hangping Z., Bin Z.** A New Learning Algorithm for Neural Networks with Integer Weights and Quantized Non-linear Activation Functions. Artificial Intelligence in Theory and Practice. Boston: Springer, 2008:427—431.

24. **Shuang Wu, Guoqi Li, Feng Chen, Luping Shi.** Training and Inference with Integers in Deep Neural Networks. Proc. VI Intern. Conf. Learning Representations. 2018:1—14.

25. **Vapnik V.N., Chervonenkis A.Ya.** Teoriya Raspoznavaniya Obrazov. M.: Nauka, 1974.

26. **Fan R.-E., Chang K.-W., Hsieh C.-J., Wang X.-R., Lin C.-J.** LIBLINEAR: a Library for Large Linear Classification. J. Machine Learning Research. 2008;9: 1871—1874.

Сведения об авторе:

Мамонтов Андрей Игоревич — кандидат технических наук, доцент кафедры математического и компьютерного моделирования НИУ «МЭИ», e-mail: MamontovAI@yandex.ru

Information about author:

Mamontov Andrey I. — Ph.D. (Techn.), Assistant Professor of Mathematical and Computer Modeling Dept., NRU MPEI, e-mail: MamontovAI@yandex.ru

Работа выполнена при поддержке: РФФИ (гранты № 19-01-00294)
The work is executed at support: RFBR (Grantst No. 19-01-00294)

Статья поступила в редакцию: *20.11.2019*
The article received to the editor: *20.11.2019*