УДК 004.021

М.А. Тищенко, Д.А. Кордубан

Эффективная аффинная аппроксимация в метрике L_1

Предложен эффективный алгоритм поиска оптимального в терминах метрики L_1 аффинного преобразования одного вектора в другой, открывающий возможности к применению метрики L_1 в задачах, где она не использовалась.

An efficient algorithm of the search for the optimal affine transformation of a vector to another one in terms of L_1 metric is suggested. The algorithm unveils the possibility of exploiting L_1 metric in problems where it was not used.

Запропоновано ефективний алгоритм пошуку оптимального в термінах метрики L_1 афінного перетворення одного вектора в інший, який дозволяє використовувати метрику L_1 в задачах, де вона не використовувалася.

Введение. Метрики L_1 и L_2 – едва ли не самые распространенные во многих областях прикладной математики, и, в частности, в структурном распознавании. Каждая из них, в контексте тех или иных задач, имеет как преимущества, так и недостатки. Один из недостатков метрики L_2, применительно к задачам аппроксимации экспериментальных данных, – существенное влияние «выбросов» на результат аппроксимации [1], в то время как метрика L_1 этого лишена. В свою очередь, преимуществом метрики L_2 в сравнении с L_1 является то, что во многих случаях задачи, сформулированные в этой метрике, аналитически разрешимы. Благодаря этому преимуществу, метрика L_2 широко применялась как на заре структурного распознавания [2, 3], так и в более поздних работах [4–6]. В частности, задача аффинной аппроксимации, т.е. минимизации функции вида:

$$\sum_{i=1}^{n} \left\| x_i - a \cdot y_i - b \right\| \qquad (1)$$

в метрике L_2 может быть решена аналитически. В то же время при постановке в метрике L_1 даже построение эффективного алгоритма решения этой задачи выглядит довольно проблематично. В данной статье не ставится и тем более не решается вопрос о том, какую из указанных метрик следует применять. В ней предлагается итеративный алгоритм решения задачи минимизации функции (1) в метрике L_1, который в наихудшем случае работает за время порядка $O(n^2)$.

Постановка и анализ задачи

Пусть задано два вектора $x = (x_1, x_2, \ldots, x_n)^T \in \in R^n$ и $y = (y_1, y_2, \ldots, y_n)^T \in R^n$. Необходимо найти параметры a и b, доставляющие минимум функции:

$$F(a,b) = \sum_{i=1}^{n} \left| x_i - a \cdot y_i - b \right|. \qquad (2)$$

Отметим, что функция (2) обладает таким свойством, что поиск минимизирующих ее параметров a и b достаточно проводить лишь по тем их значениям, которые обращают в ноль по крайней мере два слагаемых в сумме (2). Более строго это утверждение формулируется в лемме 2. Лемма 1 представляет собой вспомогательное утверждение, необходимое для доказательства леммы 2.

Лемма 1. Пусть задано два вектора $x = (x_1, x_2, \ldots, x_n)^T \in R^n$ и $y = (y_1, y_2, \ldots, y_n)^T \in R^n$, а также функция $f: R \to R$ вида:

$$f(a) = \sum_{i=1}^{n} \left| x_i - a \cdot y_i \right|.$$

Тогда существует такой индекс j и такое значение a^*, что

$$x_j - a^* \cdot y_j = 0,$$

и

$$f(a^*) = \min_a f(a).$$

Лемма 2. Пусть a^* и b^* – значения параметров a и b соответственно, при которых (2) достигает минимума. Тогда существуют такие a'и b', что для некоторых двух различных индексов j и k выполняется требование: $x_j - a' \cdot y_j - b' = 0$ и $x_k - a' \cdot y_k - b' = 0$ и соблюдается равенство $F(a^*, b^*) = F(a', b')$.

Доказательство. Предположим, что значения a^* и b^* не обращают в ноль ни одно сла-

гаемое из (2). В противном случае следует перейти к п. 2 доказательства.

Рассмотрим (2) как функцию от a при фиксированном значении $b = b^*$. В силу леммы 1 найдется такое значение a', что $F(a',b^*) = F(a',b')$ и для некоторого индекса j выполняется:

$$x_j - a' \cdot y_j - b^* = 0 .$$

Следовательно, без потери общности можно считать, что параметры a^* и b^* обращают в ноль по крайней мере одно слагаемое в (2).

Теперь мы имеем значения параметров a^* и b^*, обращающие в ноль по крайней мере одно слагаемое из суммы (2). Предположим, что такое слагаемое единственно. В противном случае требование леммы уже выполняется. Обозначим через j индекс этого слагаемого. Рассмотрим множество таких пар (a, b), при которых j-е слагаемое обращается в ноль:

$$x_j - a \cdot y_j - b = 0 . \qquad (3)$$

Это множество задается зависимостью

$$b = x_j - a \cdot y_j . \qquad (4)$$

Подставим это значение b в (2):

$$F(a, j) = \sum_{i=1}^{n} \left| x_i - x_j - a \cdot (y_i - y_j) \right| . \qquad (5)$$

В силу леммы 1 существует такое значение a', что $F(a^*, j) = F(a', j)$ и для некоторого $k \neq j$ выполняется равенство:

$$x_k - x_j - a' \cdot (y_k - y_j) = 0 .$$

Следовательно, мы нашли такие значения a' и $b' = x_j - a' \cdot y_j$, что $F(a^*, b^*) = F(a', b')$ и для некоторых двух различных индексов j и k выполняется требование: $x_j - a' \cdot y_j - b' = 0$ и $x_k - a' \cdot y_k - b' = 0$.

Лемма доказана.

Из леммы 2 непосредственно следует алгоритм поиска оптимальных значений параметров a и b, сложности $O(n^3)$. Проводится перебор по всем парам индексов i и $j, j > i$. На основании системы уравнений:

$$\left\{ \begin{array}{l} x_i - a_{ij} \cdot y_i - b_{ij} = 0 \\ x_j - a_{ij} \cdot y_j - b_{ij} = 0 \end{array} \right\} \qquad (6)$$

определяются параметры a_{ij} и b_{ij}. Из множества параметров $P = \{(a_{ij}, b_{ij}) : i = 1,...,n-1;\ j = i+1, \ldots, n\}$ путем прямого перебора выбирается пара (a_{ij}, b_{ij}), доставляющая минимум функции (2). С учетом леммы 1 это и будет решением задачи.

Множество P содержит $\frac{n^2 - n}{2}$ элементов. Для подсчета значения функции (2) при некоторых фиксированных значениях a и b необходимо порядка n операций. Таким образом, получаем сложность $O(n^3)$.

В следующем разделе указан более эффективный алгоритм решения задачи (2).

Эффективный алгоритм решения задачи

Решение задачи в одномерном случае

Рассмотрим задачу минимизации функции (5) по переменной a. В силу леммы 1 минимум этой функции достаточно искать при таких значениях параметра a, при которых по крайней мере одно из слагаемых в (5) равно нулю. Обозначим множество таких значений параметра a через $A(j) = \{a : \exists\, i : x_i - x_j - a \cdot (y_i - y_j) = 0\}$.

Запишем выражение для производной слева функции (5):

$$F'(a-0, j) = \sum_{i: x_i - x_j - a(y_i - y_j) < 0} (y_i - y_j) - \sum_{i: x_i - x_j - a(y_i - y_j) > 0} (y_i - y_j) . \qquad (7)$$

Обозначим разности $x_i - x_j$ и $y_i - y_j$ через $\tilde{x}_i$ и $\tilde{y}_i$ соответственно[1]. Перепишем в этих обозначениях выражение (7) не указывая, для удобства, зависимости от j:

$$F'(a-0) = \sum_{i: \tilde{x}_i - a \cdot \tilde{y}_i < 0} \tilde{y}_i - \sum_{i: \tilde{x}_i - a \cdot \tilde{y}_i > 0} \tilde{y}_i . \qquad (8)$$

Без потери общности предположим, что $\tilde{y}_i > 0$ для всех $i = 1,...,n$. Тогда (8) можно переписать в виде:

---

1 В случае когда требуется минимизировать не (2), а функцию вида: $F(a,b) = \sum_{i=1}^{n} |x_i - a \cdot y_i - b \cdot z_i|$, можно применить подобные рассуждения. Тогда через $\tilde{x}_i$ и $\tilde{y}_i$ следует обозначить величины: $\tilde{x}_i = x_i - \frac{x_j z_i}{z_j}, \tilde{y}_i = y_i - \frac{y_i z_i}{z_j}$.

$$F'(a-0)=\sum_{i:a>\frac{\tilde{x}_i}{\tilde{y}_i}}\tilde{y}_i-\sum_{i:a<\frac{\tilde{x}_i}{\tilde{y}_i}}\tilde{y}_i. \qquad (9)$$

При наименьших значениях a из $A(j)$ все $\tilde{y}_i, i=1,...,n$ будут содержаться во второй сумме выражения (9), и, следовательно, производная $F'(a-0)$ будет отрицательной. При увеличении a количество слагаемых во второй сумме выражения (9) будет уменьшаться, а в первой – увеличиваться. При некотором значении a' параметра a наступит одно из следующих событий:

- производная станет положительной:

$$F'(a'-0)>0, \qquad (10)$$

- производная станет равной нулю:

$$F'(a'-0)=0. \qquad (11)$$

Если имеет место первый случай, то минимум функции (5) достигается при единственном значении параметра a, а именно при $a^*=prev(a')$, где $prev(a')$ – предшествующий a' по величине элемент множества $A(j)$. Если же имеет место второй случай, то минимум достигается на интервале $[prev(a'),a']$.

На основании изложенных соображений построим алгоритм, выдающий оптимальное значение параметра a^* и множество индексов:

$$I(a^*,j)=\left\{i: x_i-x_j-a^*\cdot(y_i-y_j)=0, i\neq j\right\}. \qquad (12)$$

Предположим, что имеется алгоритм, который указывает a^*. Такой алгоритм, для случая, когда все $\tilde{y}_i$ в (9) равны единице, описан в [7] и назван в данном источнике *Select*. Его сложность составляет $O(n)$. В [7] имеется также рандомизированная версия этого алгоритма – *Randomized_Select*, для которого $O(n)$ – ожидаемое время работы. Второй из указанных алгоритмов – более практичный, так как в его асимптотической оценке скрыта меньшая константа. Оба эти алгоритма легко обобщаются на случай произвольных $\tilde{y}_i$. Опишем обобщение алгоритма *Randomized_Select*.

Алгоритм 0

Ш а г 1. Сформировать рабочий массив пар:

$$A=\left[\left(\frac{\tilde{x}_i}{\tilde{y}_i},\tilde{y}_i\right),\ i=1,...,n\right].$$

Первый элемент i-й пары назовем ключом, а второй – значением, и обозначим соответственно $k(A[i])$, и $v(A[i])$.

Задать границы обрабатываемой области массива A:

$$head:=1,\ tail:=n.$$

Ш а г 2. Случайным образом выбрать элемент q в подмассиве $A[head...tail]$. Преобразовать подмассив $A[head...tail]$ таким образом, чтобы:

$$k(A[i])\le k(q),\ head\le i\le index(q);$$
$$k(A[i])>k(q),\ index(q)<i\le tail,$$

где $index(q)$ – индекс элемента q в преобразованном подмассиве.

Ш а г 3. Подсчитать суммы элементов массива A, расположенных по левую и правую стороны от q:

$$s_l=\sum_{i=1}^{index(q)}v(A[i]),\quad s_r=\sum_{i=index(q)+1}^{n}v(A[i]).$$

Ш а г 4. Если $s_l=s_r$, завершить работу алгоритма, и в качестве ответа выдать $a^*:=k(q)$. Если $s_l<s_r$, передвинуть начало обрабатываемой области массива A:

$$head:=index(q)+1.$$

Если $s_l>s_r$, передвинуть конец обрабатываемой области массива A:

$$tail:=index(q)-1.$$

Если $head=tail$, завершить работу алгоритма, и в качестве ответа выдать $a^*:=k(A[head])$. В противном случае перейти к шагу 2.

В [7] изложено доказательство того факта, что алгоритм *Randomized_Select* работает в среднем за время $O(n)$. Это доказательство может быть механически перенесено на алгоритм 0.

На основании значения a^*, выданного алгоритмом 0, можно построить множество $I(a^*,j)$ при помощи следующего простого алгоритма.

Алгоритм 1

Ш а г 1. Получить значение a^* с помощью алгоритма 0.

Ш а г 2. Построить множество $I\left(a^*, j\right)$:

$$I\left(a^*, j\right) = \left\{ i : \frac{x_i - x_j}{y_i - y_j} = a^* \right\}. \qquad (13)$$

Решение задачи в двумерном случае

Построим алгоритм, который находит минимум функции (2) путем последовательного применения алгоритма 1 к функции (5).

Алгоритм 2

Ш а г 1. Определим множество $V = \varnothing$ индексов слагаемых из суммы (2), которые уже обработаны алгоритмом. Выберем из этой суммы какое-либо из слагаемых. Пусть оно соответствует индексу j. Добавим индекс j к множеству V.

Будем рассматривать такие значения параметров a и b, которые обращают в ноль j-е слагаемое (рис. 1):

$$x_j - a \cdot y_j - b = 0. \qquad (14)$$

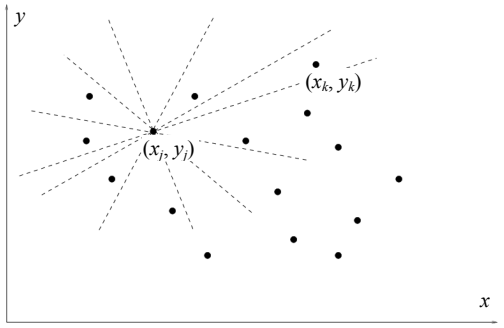

Рис. 1. Иллюстрация работы алгоритма 2. Различные прямые соответствуют различным значениям параметров a и b, обращающим в ноль j-е слагаемое из (2)

Выразив b из (14) и подставив в (2) получим (5), т.е. функцию, минимизируемую алгоритмом 1. Применив его, найдем такое значение a^* параметра a и множество $I(a^*, j)$ (рис. 2), что:

$$a^* = \arg\min_a F\left(a, j\right).$$

Ш а г 2. Выберем из множества $I(a^*, j)$ еще не обработанные элементы:

$$I\left(a\ , j\right) := I\left(a\ , j\right) \setminus \left(I\left(a\ , j\right) \cap V\right).$$

Последовательно применим алгоритм 1 к элементам множества $I(a^*, j)$, добавляя соответствующие индексы в множество V до тех пор, пока найдем такой индекс $k \in I(a^*, j)$, что:

$$\min_a F\left(a, k\right) < F\left(a^*, j\right), \qquad (15)$$

либо просмотрим все элементы множества.

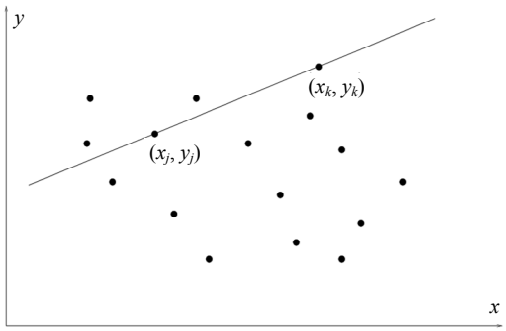

Рис. 2. Иллюстрация работы алгоритма 2. Пара точек, через которые проходит прямая, соответствует двум слагаемым, обращающимся в ноль при текущих значениях параметров a и b

Ш а г 3. Если в множестве $I(a^*, j)$ не существует индекса k, удовлетворяющего (15), то алгоритм заканчивает свою работу и выдает результат – значения параметров (a^*, b^*), $b^* = x_j - a^* \cdot y_j$. В противном случае алгоритм переходит к шагу 2 с новыми значениями a^* и j:

$$a^* := \arg\min_a F\left(a, k\right),$$

$$j := k.$$

В процессе выполнения алгоритма множество V монотонно увеличивается, следовательно алгоритм 2 закончит работу за количество шагов, не превосходящее размерности пространства, в котором определены вектора x и y.

В худшем случае алгоритм 2 выполнит n вызовов алгоритма 1, просмотрев все n возможных значений индекса j в (5). Первый шаг алгоритма 1 выполняется за время порядка $O(n)$ [7]. Второй его шаг также выполняется за линейное по n время. Таким образом, алгоритм 2 в худшем случае выполнится за $O(n^2)$.

Теорема 1. Точка останова алгоритма 2 – точка глобального минимума функции (2).

Доказательство. В силу выпуклости функции (2) ее локальный минимум – глобален. Следовательно, для доказательства теоремы достаточно показать, что точка (a^*, b^*) – локальный минимум функции (2).

Для наглядности дальнейших рассуждений воспользуемся рис. 3. Каждая из изображенных на нем прямых представляет множество точек, соответствующих таким значениям параметров a и b, которые обращают в ноль некоторое фиксированное слагаемое из суммы (2). Следовательно, точки пересечения этих прямых определяют те значения параметров (a, b), при которых два или более слагаемых из (2) обращаются в ноль. Троеточия между прямыми на рис. 3 обозначают возможное наличие, помимо P_1P_m и P_kP_u, также других прямых, пересекающихся в точке M, что соответствует той ситуации, когда в этой точке обращаются в ноль более двух слагаемых.

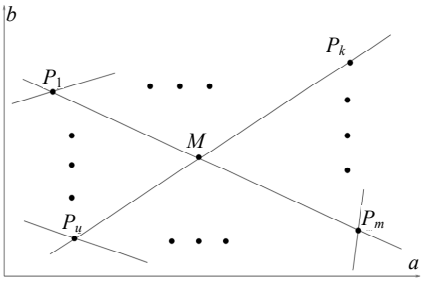

Рис. 3. Точка минимума функции (2)

Пусть количество прямых, проходящих через точку M, равняется $r/2$. Рассмотрим некоторый луч α_i, исходящий из точки M, на котором обращается в ноль одно из слагаемых в сумме (2). Возможны два случая:

- существует хотя бы одна точка P_i на этом луче, в которой более одного слагаемого из (2) обращается в ноль. Без потери общности будем считать, что на отрезке (M, P_i) обращается в ноль единственное слагаемое из суммы (2);
- на всем луче, за исключением точки M, только одно слагаемое из (2) обращается в ноль. Выберем на этом луче произвольную точку, отличную от M, и обозначим ее P_i.

Теперь точки $P_1, P_2, ..., P_k, ..., P_m, ..., P_u, ..., P_r$ выбраны таким образом, что на отрезках (M, P_i), $i = 1, \ldots, r$ обращается в ноль единственное слагаемое из суммы (2). Следовательно, на этих отрезках функция (2) – линейна.

Предположим, что алгоритм 2 закончил работу в точке M. Покажем, что в этой точке функция (2) имеет локальный минимум. В соответствии с шагами 2–3 алгоритма 2, значения функции (2) в точках $P_1, P_2, ..., P_r$ не меньше, чем в точке M, иначе алгоритм перешел бы в одну из этих точек. В силу линейности функции (2) в пределах каждого из треугольников $P_1P_2M, P_2P_3M, ..., P_{r\text{-}1}P_rM$, можно утверждать, что в каждой точке многоугольника $P_1, P_2, ..., P_r$ значение функции (2) не меньше ее значения в точке M. Следовательно, точка M – локальный минимум функции (2), а в силу ее выпуклости, и глобальный.

Теорема доказана.

Замечания относительно программной реализации

Повышая быстродействие программной реализации алгоритма 2, можно разбить слагаемые суммы (2) на N классов, в каждом из которых значения пар (x_i, y_i) одинаковы:

$$F(a,b) = \sum_{i=1}^{N} w_i \cdot |x_i - a \cdot y_i - b|,$$

где w_i – количество элементов в i-м классе.

В этом случае (9) примет вид:

$$F'(a-0) = \sum_{i:a>\frac{\tilde{x}_i}{\tilde{y}_i}} w_i \cdot \tilde{y}_i - \sum_{i:a<\frac{\tilde{x}_i}{\tilde{y}_i}} w_i \cdot \tilde{y}_i. \quad (16)$$

На первом шаге алгоритма 0 массив A следует формировать с учетом весовых коэффициентов w_i:

$$A = \left\{ \left(\frac{\tilde{x}_i}{\tilde{y}_i}, w_i \cdot \tilde{y}_i \right), i = 1, ..., N \right\}.$$

В случае когда $N \ll n$, такой прием позволяет существенно сократить количество вычислений.

Таким образом, в статье предложен эффективный алгоритм минимизации функции

$$F(a,b) = \sum_{i=1}^{n} |x_i - a \cdot y_i - b|. \quad (17)$$

Заключение. Алгоритм минимизации функции открывает возможность для использования метрики L_1 при постановке задач, где она ранее не участвовала из-за отсутствия эффективных

алгоритмов минимизации функции (17). К таким задачам можно отнести немалое количество прикладных проблем распознавания образов, и, в частности, обработки и распознавания изображений.

1. *Стецюк П.И.*, *Колесник Ю.С.*, *Лейбович М.М.* О робастности метода наименьших модулей // Компьютерная математика. – 2002. – **2**. – С. 114–123.
2. *Ковалевский В.А.* Корреляционный метод распознавания изображений. // Ж. выч. математикиии мат. физика. – 1962. – **2**, № 4.– С. 684–689.
3. *Ковалевский В.А.* Алгоритм разделения машинописной строки на знаки при отсутствии пробелов // III Всесоюз. конф. по информационно-поисковым системам и автоматизированной обработке науч.-техн. информации. – 1967. –Т. 3. – С. 156–164.
4. *Рябоконь Д.И.* Пространственная реконструкция поверхностей по стереопаре изображений с помощью алгоритмов поиска минимального сечения на графе // УСиМ. –2004. – № 3. – С. 47–51.
5. *Рябоконь Д.И.* Восстановление пространственной конфигурации объектов и сцен по их стереоизображениям // Там же. – 2005. – № 1. – С. 22–31.
6. *Система* доступу до приміщення на основі розпізнавання людських облич / В.М. Кийко, К.В. Кийко, В.В. Мацелло та ін. // Пр. Восьмої Всеукр. міжнар. конф. «УкрОБРАЗ'2006», Київ, 28–31 серпня 2006. – С. 123–126.
7. *Алгоритмы*: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест и др. – М.: Вильямс, 2007. – 1296 с.

Поступила 16.03.2010
Тел. для справок: (044) 502-6314 (Киев)
E-mail: *maxim.tischenko@gmail.com*

●

Окончание статьи Г.А. Донца и др.

9 (7, 2, 1, 3, 8); 15 (8, 3, 1, 4, 9); 19 (8, 6, 2, 7, 9); 22 (7, 3, 1, 4, 8); 31 (8, 5, 2, 6, 9); 34 (9, 7, 5, 2, 6, 8); 37 (8, 6, 5, 7, 9); 38 (9, 6, 3, 1, 4, 7); 39 (6, 3, 1, 4, 7).

На каждой из этих цепей необходимо построить правильную нумерацию $f_5(3)$ или $f_6(3)$, что выполняется достаточно легко. Для деревьев 21 и 33 правильная нумерация приведена на рис. 7.

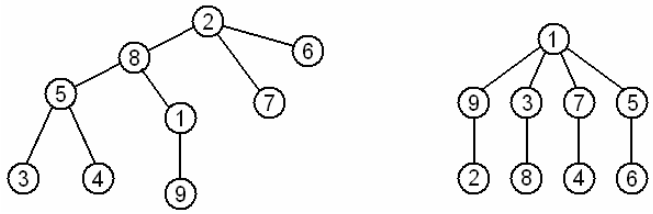

Рис. 7. Правильная нумерация графов 21 и 33

Этот метод можно применять и для деревьев более высокого порядка. На определенном уровне придется прибегнуть к помощи вычислительной техники. Как показала практика, непосредственное использование вычислительной техники для простого перебора вариантов уже для деревьев порядка 20–22 наталкивается на непреодолимые препятствия технического характера.

Заключение. В данной статье продемонстрировано решение проблемы Роса для деревьев с числом вершин $n = 9$. Очевидно, что этот же метод можно распространить и на деревья более высоких порядков. Для этого понадобится найти правильную нумерацию для новых, более сложных, чем звезда, цепь и гусеница, конфигураций. Об этом пойдет речь в последующих публикациях.

1. *Петренюк А.Я.* Півобертові деревні факторизації повних графів // Укр. матем. журнал. – 2001. – **53**, № 5. – С. 710–716.
2. *Харари Ф.* Теория графов. – М.: Мир, 1973. – С. 267–268.
3. *Rosa A.* On certain valuations of the vertices of a graph. – New York: Gordon and Breach, 1967. – P. 349–355.
4. *Cahit A.* On graceful trees // Bull. of the ICA. – 1994. – 12 Sept. – P. 15–18.
5. *Донец Г.А.* Об одной задаче нумерации вершин деревьев // Матем. машины и системы. – 2010. – № 1. – С. 17–24.

Поступила 19.02.2010
Тел. для справок: (044) 526-2188 (Киев)

●