



БИБЛИОТЕКА
ТЕХНИЧЕСКОЙ
КИБЕРНЕТИКИ

**АЛГОРИТМЫ
ОБУЧЕНИЯ
РАСПОЗНАВАНИЮ
ОБРАЗОВ**

• СОВЕТСКОЕ РАДИО •

АЛГОРИТМЫ ОБУЧЕНИЯ РАСПОЗНАВАНИЮ ОБРАЗОВ

Под редакцией В. Н. ВАПНИКА



МОСКВА
«СОВЕТСКОЕ РАДИО» 1973

6Ф01

В 17

УДК 62—5:007:621.391:519.2

Члены редакционного совета:

Трапезников В. А. (председатель), Челюсткин А. Б. (зам. председателя), Бусленко Н. П., Виленкин С. Я., Воронов А. А., Гаазе-Рапопорт М. Г., Дудников Е. Г., Ицкович Э. Л., Круг Г. К., Мамионов О. Г., Осколков И. О., Пархоменко П. П., Пинскер М. С., Плискин Л. Г., Поспелов Г. С., Райбман Н. С., Самойленко С. И., Таль А. А., Флейшман Б. С., Хургин Я. И., Якобсон Б. М.

Алгоритмы обучения распознаванию образов. Под ред. В. Н. Вапника М., «Сов. радио», 1973 г. 200.

В книге содержится подробное описание алгоритмов, успешно справляющихся с решением практических задач, и программ, реализующих эти алгоритмы. Особенностью книги является широта охвата конструктивных идей построения алгоритмов распознавания образов.

Книга рассчитана на широкий круг читателей, работающих в области практического применения теории, а также специалистов, занятых построением теории распознавания образов. Она окажется также полезной для студентов вузов и аспирантов.

27 рис., 1 табл. 28 назв.

АЛГОРИТМЫ ОБУЧЕНИЯ РАСПОЗНАВАНИЮ ОБРАЗОВ

Под редакцией В. Н. Вапника

Редактор М. С. Гордон

Художественный редактор З. Е. Вендрова

Технический редактор З. Н. Ратникова

Корректоры Е. П. Озерецкая, З. Г. Галушкина



Scan AAW

Сдано в набор 23/III 1972 г.

Подписано в печать 13/III 1973 г. Т-00492

Формат 84×108¹/₃₂ Бумага машиномелованная

Объем 10,5 усл. п. л. 10,439 уч. изд. л.

Тираж 6400 экз. Зак. 1366 Цена 55 коп.

Издательство „Советское радио“, Москва, Главпочтамт, п/я 693.

Московская типография № 10 Союзполиграфпрома при Государственном комитете Совета Министров СССР по делам издательств, полиграфии и книжной торговли. Москва, М-114, Шлюзовая наб., 10.

В 3314-041
046(01)-73 103-72

ПРЕДИСЛОВИЕ

Задача обучения распознавания образов была сформулирована в конце 50-х годов и до сих пор остается одной из основных задач кибернетики. Это объясняется не только тем, что методы обучения распознаванию образов являются основой для решения многих практических задач, но и самим математическим содержанием задачи. Задача обучения распознаванию образов является по существу задачей о восстановлении простой функциональной зависимости, представляющей собой характеристическую функцию, т. е. функцию, которая принимает только два значения — нуль или единица.

Эта особенность задачи восстановления оказалась решающей: впервые удалось действительно эффективно восстанавливать многомерные функции. При этом удивительным оказалась «удачливость» задач обучения распознаванию образов: восстанавливались многомерные зависимости по выборкам настолько малого объема, что успех восстановления функции невозможно объяснить с помощью существующей статистической теории.

Неожиданным было и то, что алгоритмы обучения распознаванию образов, основанные на хорошо разработанных методах параметрической статистики на практике оказались недостаточно эффективными.

Поэтому задача обучения распознаванию образов решается обычно как задача непараметрической статистики, теория которой еще только разрабатывается.

Таким образом, построение алгоритмов обучения распознаванию образов как правило теоретически опирается на статистическую теорию восстановления функциональных зависимостей.

Это породило большое количество экспериментальных исследований: создавались и создаются различные алгоритмы обучения распознаванию образов, часть из которых успешно справляется с решением практических задач.

Издание книги преследует две цели. Во-первых, познакомить читателей с различными конструктивными идеями алгоритмов, успешно решающих задачи распознавания. Во-вторых, дать возможность читателю, интересующемуся вопросами приложений, получить с минимальной затратой сил программы, реализующие эффективные алгоритмы обучения распознаванию образов.

В. ВАПНИК

Москва, май 1971 г.

В. Н. Вапник

Машины, обучающиеся распознаванию образов

Задача обучения распознаванию образов занимает особое место в кибернетике: это задача, в которой строгий математический подход тесно переплетается с эвристическим.

Поэтому естественно возникают недоразумения: с одной стороны, попытка создать программу, способную обучиться широкому классу задач на основе математического анализа, не всегда приводит к цели; с другой стороны, во многих эвристических программах могут быть обнаружены нарушения общих принципов построения алгоритма, что приводит к неудовлетворительной работе программы.

До сих пор остается дискуссионным вопрос, является ли задача обучения распознаванию образов задачей математического анализа или эвристического программирования. Тем не менее строгая постановка этой задачи существует.

Рассмотрим сначала ее частную постановку.

1. ЧАСТНАЯ ПОСТАНОВКА ЗАДАЧИ ОБУЧЕНИЯ РАСПОЗНАВАНИЮ ОБРАЗОВ

Формализуется постановка следующей задачи. Учитель делит возникающие перед ним ситуации на некоторое небольшое число классов. Требуется создать такую машину^{*)}, которая после наблюдения за действиями учителя классифицировала бы ситуации примерно так же, как и он.

Наиболее просто формализуется слово ситуация. Под ним понимается набор чисел, однозначно соответствующих тому явлению или предмету, которые подлежат классификации.

^{*)} В дальнейшем под термином машина будем понимать программу, реализующую определенный алгоритм.

Так, в задаче классификации геологических пластов эти числа соответствуют различным параметрам пласта, в задаче классификации изображений числа кодируют степень зачерненности растра, в задаче медицинской диагностики — различные анализы больного, анамнез и т. д.

Формально каждая ситуация — это точка в евклидовом пространстве E_n . Множество всех возможных ситуаций может совпадать со всем пространством E_n или не совпадать с ним (т. е. не все сочетания параметров возможны).

Учитель делит все ситуации на m классов. Для простоты будем считать, что $m=2$ (такое допущение не принципиально потому, что последовательным делением ситуаций на два класса можно получить деление на $m > 2$ классов). Если учитель относит некоторую ситуацию к первому классу, то этот факт условимся отмечать цифрой 1, если же он относит ситуацию ко второму классу — цифрой 0. Пусть учитель осуществляет классификацию с помощью характеристической функции g_0 .

Задача состоит в том, чтобы, наблюдая некоторое число пар: ситуация x — ее классификация g_0x , найти функцию, близкую в определенном смысле к характеристической функции, используемой учителем, т. е. восстановить функцию g_0x по ее значениям в заданных точках.

Задача восстановления функции по ее значениям является традиционной задачей математики. При ее постановке необходимо определить:

- 1) в каком смысле восстанавливается функция (т. е. что значит функция, близкая к заданной);
- 2) в каком классе функций ведется восстановление;
- 3) как задаются точки, в которых определена функция.

Конкретное содержание понятия близости, способ задания класса функций и наконец особенности закона задания точек, в которых определена функция, и составляют особенность задачи обучения распознаванию образов.

1. В задаче обучения распознаванию образов ситуации, поступающие для распознавания, возникают случайно и независимо, согласно некоторому распределению $P(x)$ (само распределение неизвестно, но важно, что оно существует и фиксировано). Будем считать, что две характеристические функции: функция $A(x)$ и функция $B(x)$ близки, если вероятность того, что они по-разному

классифицируют возникшую ситуацию, мала, т. е. если справедливо неравенство

$$\int_X |A(x) - B(x)| P(x) dx < \varepsilon, \quad (1)$$

где ε — заданное число.

2. В частной постановке задачи требуется, чтобы в том классе функций G , в котором ведется восстановление, содержалась характеристическая функция g_0 , используемая учителем.

3. Эта функция g_0 определена в l точках, каждая из которых выбирается случайно и независимо, согласно фиксированному распределению $P(x)$.

Пусть задан алгоритм восстановления функции в классе G . Тогда будет ли найдена в классе G функция, близкая к функции, используемой учителем, зависит от того, в каких точках она была определена. При одних наборах из l точек будет восстановлена функция, близкая к функции, используемой учителем, при других наборах этого не произойдет.

Для функции g_0 , используемой учителем, рассмотрим множество $X^l(g_0)$, каждый элемент которого есть набор из l точек, неблагоприятных для заданного алгоритма восстановления. Если на множестве X определена вероятностная мера, то она индуцирует меру и на множестве $X^l(g_0)$.

Задача состоит в том, чтобы для заданного класса функций G построить такой алгоритм восстановления функции g_0 для которого вероятностная мера неблагоприятного набора из l точек была бы меньше $\eta > 0$.

В задаче обучения распознаванию образов принята следующая терминология. Характеристическая функция g называется *решающим* правилом, соответственно класс G — классом решающих правил; набор из l точек и значений функции g в этих точках — *обучающей последовательностью* длины l , алгоритм восстановления характеристической функции g — *обучающейся машиной*.

Используя эти термины, задача состоит в том, чтобы для заданного класса решающих правил G построить такую обучающуюся машину, которая по обучающей последовательности длины l выбрала бы решающее правило, которое с вероятностью $1 - \eta$ было бы близко к решающему правилу, используемому учителем, т. е. $g_0 < G$.

Такова формальная сторона частной постановки задачи обучения распознаванию образов.

Неформальная сторона заключается в том, чтобы суметь выбрать класс решающих правил G , адекватный тем задачам, которые предстоит решать машине. Выбор того или иного класса решающих правил никак не может быть сколько-нибудь обоснован средствами математического анализа и потому класс решающих правил задается исходя из чисто эвристических соображений.

2. КЛАССИЧЕСКАЯ СХЕМА ВОССТАНОВЛЕНИЯ ФУНКЦИИ

Классическая схема восстановления функции $g_0 \in G$ заключается в следующем. В классе функций G выбирается функция g^* , значения которой в заданных точках совпадают со значениями функции g_0 . В нашем случае это означает, что в классе G выбирается такая характеристическая функция, которая классифицирует ситуации обучающей последовательности точно так, как учитель (так как функция g_0 заведомо принадлежит классу G , то такие функции в классе существуют и при восстановлении может быть выбрана любая из них).

Спрашивается, способна ли к обучению машина, реализующая классическую схему восстановления функции.

Ответ на этот вопрос дает следующая теорема, приведенная в работе [1].

Теорема 1

Если класс функций G состоит из N различных на множестве G функций^{*)}, то обучающаяся машина, реализующая классическую схему восстановления функции, способна обучаться, используя обучающую последовательность длины:

$$l = \frac{\log N - \log \eta}{-\log(1 - \epsilon)}. \quad (2)$$

Требование, чтобы класс G состоял из конечного числа различных на множестве X функций, не является серьезным ограничением. Так, если входная ситуация x

^{*)} Две функции $A(x)$ и $B(x)$ различны на множестве X , если существует такое $x \in X$, что $A(x) \neq B(x)$. Класс функций G состоит из N различных на множестве X функций, если из этого класса может быть выделено не более N функций, различных на множестве X .

есть бинарный вектор (т. е. вектор, координаты которого могут принимать только два значения — «0» или «1»), то возможно всего 2^{2^n} преобразований (по числу всех булевых функций размерности n).

Согласно формуле (2), чем больше N , тем большей, вообще говоря, должна быть длина обучающей последовательности. Это хорошо согласуется с интуитивными представлениями: чем меньше имеется априорных сведений о классе решающих правил, которому принадлежит решающее правило учителя, тем шире следует его оговаривать и тем большей должна быть длина обучающей последовательности. На основе этой формулы можно сделать вывод: чем «универсальнее» обучающая машина, т. е. чем большему количеству задач эта машина может обучиться, тем дольше она учится.

Следует отметить, что хотя оценка (2) не улучшаема *), при решении практических задач она часто оказывается сильно завышенной, так как в ней использовались только сведения о числе различных на множестве X функций в классе G и никак не учитывались ни свойства самих функций, ни особенности распределения $P(x)$. Ниже будет приведена более тонкая оценка.

В заключение рассмотрим пример машины, способной обучиться распознаванию образов. Машина предназначена для обучения классификации ситуаций, заданных бинарными векторами размерности n .

Класс операторов машины задан в виде

$$g(x) = \theta(\sum \alpha_i x_i + \alpha_0), \quad (3)$$

где

$$\theta(z) = \begin{cases} 0, & z \leq 0, \\ 1, & z > 0; \end{cases}$$

x_i — координаты вектора x ; α_i — константы, которые могут принимать любые значения.

Иначе говоря, машина рассчитана на решение таких задач, для которых априори известно, что учитель при классификации пользуется следующим правилом: выясняет, по какую сторону от гиперплоскости в евклидовом пространстве E_n попадает ситуация x и в зависимости от этого относит ее к тому или иному классу (конечно, ма-

*) Оценка считается неулучшаемой, если существуют примеры, где она достигается.

шине неизвестно, какой именно гиперплоскостью пользуется учитель. Это должно быть выяснено в ходе обучения). Машина восстанавливает решающее правило следующим образом: в ходе обучения машина строит гиперплоскость, которая делит ситуации обучающей последовательности так же, как и учитель. Затем она будет относить всякую предъявленную ситуацию к одному из двух классов, в зависимости от того, по какую сторону от построенной гиперплоскости находится эта ситуация.

Определим, способна ли такая машина к обучению и какова для нее достаточная длина обучающей последовательности. Для этого оценим число гиперплоскостей, различных на множестве бинарных векторов. Количество гиперплоскостей равно числу различных способов разделения бинарных векторов на два класса с помощью гиперплоскостей.

Число таких способов N оценено в работах [1, 2]: ($N \leq 2^{n^2}$), откуда согласно теореме 1 машина способна к обучению и достаточная для нее длина обучающей последовательности не превосходит

$$l = \frac{n^2 - \log_2 \eta^*}{-\log_2 (1 - \epsilon)}.$$

3. ОБЩАЯ ПОСТАНОВКА ЗАДАЧИ ОБУЧЕНИЯ РАСПОЗНАВАНИЮ ОБРАЗОВ

Существенным недостатком частной постановки задачи обучения распознаванию образов является требование: характеристическая функция учителя g_0 должна содержаться в классе функций G .

Общая постановка направлена на то, чтобы снять это требование. Обобщим постановку задачи восстановления функции, рассмотренную в § 1.

1. Пусть на множестве X задана вероятностная мера $\mu(x)$ и предположим, что нижняя грань вероятности разных значений характеристической функции g_0 и функций класса G равна Θ , т. е.

$$\Theta = \inf_{g \in G} \int |gx - g_0x| d\mu(x). \quad (4)$$

*) Заметим, что здесь оценка завышена. Известна оценка [3]

$$l = \frac{n \ln n - \ln \eta}{-\ln (1 - \epsilon)}.$$

Две функции g и g_0 считаются близкими в широком смысле, если вероятностная мера различных значений этих функций меньше чем $\Theta + \varepsilon$ ($\varepsilon > 0$), т. е. функция g_0 и g близки в широком смысле, если справедливо неравенство

$$\int_x |g_0 x - g x| d\mu(x) < \Theta + \varepsilon.$$

2. Класс функций G , в котором ведется восстановление, никак специально не оговаривается.

3. Функция восстанавливается по заданным в l точках значениям (точки задаются случайно и независимо).

Очевидно, что такая постановка задачи о восстановлении функции является обобщением постановки, рассмотренной выше.

В самом деле, пусть $g_0 \in G$, т. е. удовлетворено 2-е условие частной постановки. Тогда $\Theta = 0$ и понятие близости в широком смысле совпадает с понятием близости, введенным ранее. Условия 3 в обеих постановках идентичны.

В терминах задачи обучения распознаванию образов общая постановка формулируется так.

Требуется для заданного класса решающих правил G построить такую обучающуюся машину, которая для любой функции g_0 и любой вероятностной меры $\mu(x)$ по обучающей последовательности длины l выбрала бы решающее правило $g \in G$, с вероятностью, не меньшей $1 - \eta$, близкое в широком смысле к решающему правилу, используемому учителем g_0 .

Формально постановка задачи обучения распознаванию образов не изменилась. Изменился лишь смысл понятия «близкое к g_0 решающее правило». Тем не менее новое определение близости существенно усложняет задачу. Это сказывается в первую очередь в том, что становится неприемлемой классическая схема восстановления функции: в классе G может не оказаться функции, которая в заданных точках принимает заданные значения.

4. ОБОБЩЕНИЕ КЛАССИЧЕСКОЙ СХЕМЫ ВОССТАНОВЛЕНИЯ ФУНКЦИЙ

Обобщение классической схемы заключается в том, что в классе G выбирается такая функция, которая не совпадает со значениями функции g_0 в минимальном

числе заданных точек. Эта схема восстановления содержит классическую схему: если $g_0 \subset G$, то минимальное по классу G число несовпадающих значений функций g_0 и g в заданных точках равно нулю, т. е. реализуется классическая схема восстановления функции. Возникает вопрос, обеспечивает ли реализация обобщенной классической схемы восстановления функции построение машины, способной обучиться распознаванию образов.

Формально ответ на этот вопрос должен установить, когда выбор функций с наилучшей эмпирической оценкой обеспечивает (с заданной надежностью) выбор функций с высокой истинной оценкой.

В самом деле, если функция g_0 определена в l точках, то множество функций класса G может быть разбито на $(l+1)$ непересекающееся подмножество; в подмножество G_i входят такие функции из класса G , значения которых не совпадают с функцией g_0 во всех l заданных точках; в подмножество G_i входят функции, значения которых не совпадают с функцией g_0 в i точках ($i=0, 1, \dots, l-1$).

Поставим в соответствие каждой функции подмножества G_i число $v_i = i/l$. Это число есть частота несовпадения значений функции класса G_i со значениями функции g_0 , вычисленная в заданных l точках. Согласно рассмотренной схеме восстановления функции будет выбрана функция, которой соответствует меньшая частота v_i .

Вопрос заключается в том, можно ли с заданной вероятностью утверждать, что функция с наилучшей эмпирической оценкой v_i имеет вероятностную оценку, меньшую чем $\Theta + \varepsilon$.

Обозначим для каждой функции $g \subset G$ частоту несовпадения с g_0 в l точках и вероятность несовпадения соответственно через $v_g(l)$ и p_g .

Согласно классической теореме Бернулли при неограниченном увеличении числа испытаний частота появления события стремится к вероятности (по вероятности).

Известна оценка:

$$p(|v_g - p_g| \geq \varepsilon) < e^{-\varepsilon^2 l / 2}. \quad (5)$$

Отсюда для любого фиксированного l может быть вычислена вероятность $(1-\eta)$ выполнения неравенства $|v_g - p_g| \geq \varepsilon$:

$$1 - \eta = 1 - e^{-\varepsilon^2 l / 2}.$$

Однако для того чтобы с вероятностью, не меньшей $(1-\eta)$, утверждать, что функция с минимальной эмпирической оценкой имеет вероятностную оценку, меньшую чем $(\Theta + \varepsilon)$, выполнение неравенства (5) отдельно для каждой функции g недостаточно.

Необходимо выполнение этого неравенства одновременно для всех функций $g \in G$, т. е. выполнение неравенства

$$p(\sup_{g \in G} |v_g - p_g| \geq \varepsilon) < f(\varepsilon, l), \quad (6)$$

где $\lim_{l \rightarrow \infty} f(\varepsilon, l) = 0$; ε — фиксированное число.

В случае, когда выполняется неравенство (6), говорят, что имеет место равномерная сходимость (по вероятности) частот появления событий к их вероятностям по классу G [3].

С помощью неравенства (6) определяется, какова должна быть длина обучающей последовательности l , чтобы с вероятностью, не меньшей $(1-\eta)$, можно было утверждать, что найденная по минимальной эмпирической оценке функция имеет истинную оценку, меньшую чем $(\Theta + \varepsilon)$. Для этого достаточно решить уравнение $f(\varepsilon, l) = \eta$ относительно l .

Таким образом, вопрос об эффективности рассмотренной схемы восстановления функций связан с существованием равномерной сходимости частот появления событий к их вероятностям по классу G .

Такая сходимость имеет место, когда число N различных на множестве X функций в классе G конечно. Это вытекает из того, что число функций в классе G равно N , а для каждой функции этого класса справедливо неравенство (5). Поэтому справедлива оценка:

$$p(\sup_{g \in G} |v_g - p_g| \geq \varepsilon) < Ne^{-\varepsilon^2 l / 2}, \quad (7)$$

где $e^{-\varepsilon^2 l / 2}$ — оценка вероятности из неравенства (5).

Оценка длины обучающей последовательности для машины, реализующей рассмотренную схему, определяется из равенства $Ne^{-\varepsilon^2 l / 2} = \eta$:

$$l = 2(\ln N - \ln \eta) / \varepsilon^2. \quad (8)$$

5. ПРИМЕНЕНИЕ ТЕОРИИ РАВНОМЕРНОЙ СХОДИМОСТИ ЧАСТОТ ПОЯВЛЕНИЯ СОБЫТИЙ К ИХ ВЕРОЯТНОСТЯМ В ЗАДАЧЕ ОБУЧЕНИЯ РАСПОЗНАВАНИЮ ОБРАЗОВ

Пусть задана последовательность элементов множества X длины l : x^1, \dots, x^l .

Обозначим через $\Delta^G(x^1, \dots, x^l)$ число различных на этой последовательности характеристических функций класса G .

Введем две функции:

$$m^G(l) = \max_{x^1, \dots, x^l} \Delta^G(x^1, x^2, \dots, x^l)$$

и

$$M^G(l) = E \lg \Delta^G(x^1, x^2, \dots, x^l).$$

Функция $m^G(l)$ для каждого l определяет максимальное число различных на последовательности длины l характеристических функций класса G . Назовем ее *функцией роста класса* решающих правил G . Функция $M^G l$ — математическое ожидание $\lg \Delta^G(x^1, x^2, \dots, x^l)$.

Справедливы следующие три теоремы [3].

Теорема 2

Функция $m^G(l)$ либо тождественно равна 2^l , либо мажорируется со степенной функцией $m^G(l) \leq l_k + 1$, где k — значение l , для которого впервые нарушено равенство $m^G(k) = 2^k$.

Теорема 3

Для того чтобы имела место равномерная сходимость (по вероятности) частот к вероятности по классу G , достаточно выполнение условия

$$m^G(l) \leq l^k + 1.$$

При этом для $l > 1/\varepsilon$ справедлива оценка

$$p(\sup_{g \in G} |v_g - p_g| \geq \varepsilon) < m^G(2l) e^{-\varepsilon^2 l / 8} < (2l)^k e^{-\varepsilon^2 l / 8}. \quad (9)$$

Теорема 4

Для того чтобы имела место равномерная сходимость (по вероятности) частот к вероятности по классу G , не-

обходимо и достаточно выполнение условия

$$\lim_{l \rightarrow \infty} [M^G(l)]/l = 0.$$

Теорема 2 указывает конструктивный способ оценки функции роста $m^G(l)$: достаточно установить минимальное число ситуаций, которые с помощью характеристических функций G уже не могут быть разделены на два класса всеми возможными способами. Предположим, что число таких ситуаций равно k . Тогда $m^G(l) \leq l^k + 1$, и в этом случае будем говорить, что емкость класса решающих правил ограничена. При этом имеет место равномерная сходимость частот появления событий к их вероятностям по классу G (теорема 3).

Используя неравенство (9), можно оценить достаточную длину обучающей последовательности для машин с ограниченной емкостью класса решающих правил, реализующих рассматриваемую схему восстановления функции.

Достаточная длина l^* находится из условия

$$(2l)^k e^{\epsilon^2 l/8} = \eta.$$

Она равна

$$l^* = \frac{16k}{\epsilon^2} \left(\ln \frac{16k}{\epsilon^2} - \ln \frac{\eta}{4} \right). \quad (10)$$

Достаточное условие равномерной сходимости (теорема 3) не зависит от свойств распределения $P(x)$. Необходимое и достаточное условие оказывается уже зависящим от $P(x)$ (теорема 4).

Если $\lim_{l \rightarrow \infty} [M^G(l)/l] = 0$, то машина, реализующая обобщенную схему восстановления функции, выберет нужное решающее правило (с вероятностью, не меньшей $1 - \eta$), используя конечное число примеров.

С точки зрения приведенных теорем все обучающиеся машины, реализующие описанный алгоритм выбора решающего правила, делятся на два типа. К первому типу относятся машины с ограниченной емкостью класса функций G . Такие машины выбирают нужный оператор независимо от свойств распределения $P(x)$ и для них заранее можно определить достаточную длину обучающей последовательности.

Ко второму типу относятся машины с неограниченной емкостью класса функций. Эти машины выберут нужную функцию только при благоприятной функции распределения $P(x)$. И так как при решении практических задач невозможно априори выяснить «благоприятность» функции распределения, то для таких машин до начала обучения нельзя определить ни достаточной длины обучающей последовательности, ни вероятности выбора нужной функции.

Пример 1 (обучающаяся машина первого типа). Класс функций машины задан в виде

$$gx = \Theta(\sum \alpha_i x_i + \alpha_0),$$

где x_i — координаты вектора x (не обязательно бинарные); α_i — константы, которые могут принимать любые значения.

В ходе обучения машина строит гиперплоскость, которая разделяет ситуации обучающей последовательности на два класса так, чтобы по одну сторону гиперплоскости, по возможности, находились ситуации, отнесенные учителем к первому классу, а по другую — ко второму. При этом число ситуаций, «ошибочно» попадающих в противоположное полупространство, должно быть минимально.

После обучения машина относит предъявленную ситуацию к одному из двух классов (в зависимости от того, по какую сторону от построенной гиперплоскости находится эта ситуация). Для такой машины функция роста класса G оценивается $[1, 2]$ как

$$m^G(l) \leq l^n + 1,$$

где n — размерность пространства E_n .

Согласно теореме 3 и оценке (9) эта машина на последовательности длины l^* выбирает функцию, которая близка (в широком смысле) с вероятностью, не меньшей $(1-\eta)$, к характеристической функции, используемой учителем.

Пример 2 (обучающаяся машина второго типа). В ходе обучения машина запоминает все векторы обучающей последовательности первого и второго классов. При классификации предъявленного вектора x она выясняет расстояния от этого вектора до ближайших из запомненных векторов первого и второго классов. Вектор

относится к тому классу, расстояние до ближайшего представителя которого минимально.

У этой машины емкость класса операторов G неограничена (возможно любое разделение на два класса). Поэтому успех обучения гарантирует только благоприятная функция распределения.

6. ТЕОРИЯ И ЭВРИСТИКА ПРИ ПОСТРОЕНИИ МАШИН, ОБУЧАЮЩИХСЯ РАСПОЗНАВАНИЮ ОБРАЗОВ

Согласно теоремам, рассмотренным выше, способность машин к обучению обеспечивается выполнением двух условий:

1. Емкость класса решающих правил машины ограничена.

2. В процессе обучения машина выбирает решающее правило, которое при классификации обучающей последовательности обеспечивает минимальное число ошибок.

Эти условия являются формальными требованиями к машинам, обучающимся распознаванию образов.

Неформальное требование состоит в том, чтобы в заданном классе решающих правил имелось «достаточно хорошее» правило.

Выбор класса решающих правил осуществляется эвристически. При построении обучающихся машин емкость класса решающих правил оценивается достаточно просто (см. теорему 2).

Значительно сложнее построить алгоритм выбора решающего правила, удовлетворяющий условию 2. Такой алгоритм, как правило, эквивалентен алгоритму поиска максимума многоэкстремальной функции. Обычно этот алгоритм — эвристический.

Для практического использования обучающейся машины очень важно, чтобы машина могла обучаться на обучающей последовательности сравнительно небольшой длины. Выше приведены оценки длин обучающей последовательности (2), (10). Оценки зависят как от емкостных особенностей класса решающих правил машины (или числа различных на множестве X функций), так и от требований к качеству обучения.

Эти оценки получены исходя из условия наиболее неблагоприятного распределения $P(x)$, поэтому они, как правило, оказываются завышенными.

Однако вид оценок:

$$l = \varphi_1(\varepsilon) \ln N + \varphi_2(\varepsilon, \eta), \quad (11)$$

$$l = k \ln k \cdot \bar{\varphi}_1(\varepsilon) + \bar{\varphi}_2(\varepsilon, \eta) \quad (12)$$

довольно точно отражает существо дела.

В формулах (2), (10) функции $\varphi_1(\varepsilon)$; $\varphi_2(\varepsilon, \eta)$; $\bar{\varphi}_1(\varepsilon)$; $\bar{\varphi}_2(\varepsilon, \eta)$ зависят от распределения P , в то время как сомножители $\ln N$ и $k \ln k$ — от емкости класса функций G .

При построении обучающихся машин следует учитывать, что достаточная длина обучающей последовательности прямо пропорциональна логарифму емкостных показателей класса решающих правил:

$$\text{для оценки (2)} \quad l \sim C \ln N; \quad (11a)$$

$$\text{для оценки (10)} \quad l \sim Ck \ln k. \quad (12a)$$

На практике в большинстве задач распределение достаточно благоприятно, поэтому практически коэффициент C можно принять равным 1—2.

Таким образом, емкость класса решающих функций должна быть такой, что: $\ln N \approx 100 \div 200$, либо $k \ln k \approx \approx 100 \div 200$.

Эти соображения хорошо иллюстрируются примером 2 (см. § 5). В самом деле, для небинарных векторов емкость класса решающих правил неограничена, и поэтому равномерная сходимость, а тем более способность к обучению на малом числе примеров, может обеспечить лишь весьма благоприятное распределение.

Если же векторы бинарны, то число различных на множестве бинарных векторов решающих правил конечно и равно 2^{2^n} . Формально здесь имеет место равномерная сходимость. Однако длина обучающей последовательности согласно (11a) равна

$$l \sim C \ln 2^{2^n} = C 2^n.$$

При достаточно большом n ($n > 15$) только при очень благоприятном распределении коэффициент C будет настолько мал, что длина обучающей последовательности станет практически приемлемой.

Наиболее сложным в построении машин, обучающихся распознаванию образов, является удовлетворение неформальному требованию.

По существу, задание класса решающих правил, удовлетворяющего неформальному требованию, эквивалентно указанию априорных сведений о тех задачах, которые предстоит решать машине.

К сожалению, в настоящее время неизвестны какие-либо серьезные соображения о преимуществах одного класса решающих правил перед другим. А между тем эта чисто эвристическая проблема является наиболее важной в задаче обучения распознавания образов.

Таким образом, при построении машины, обучающейся распознаванию образов, оказывается, что почти все основные моменты синтеза решаются эвристически. Поэтому задачу построения обучающейся машины можно считать задачей эвристического программирования. Однако тезис — задача построения обучающейся машины должна решаться средствами эвристического программирования — является опасным.

Хотя существуют сотни эвристических программ, но лишь немногие из них способны успешно решать задачи обучения распознаванию образов. С нашей точки зрения при построении обучающихся машин должен быть осуществлен принцип «удовлетворить неформальным и формальным требованиям средствами эвристического программирования».

В настоящей книге приведены обучающиеся программы, которые успешно решают задачи обучения распознаванию образов. Большинство из них — эвристические. И оказалось, что почти у всех этих программ емкость класса решающих правил разумно ограничена, а эвристика при выборе решающих правил направлена на удовлетворение второго формального требования.

Ниже будут разобраны все программы, приведенные в книге. С точки зрения принципа удовлетворения формальным требованиям они почти эквивалентны. Поэтому при выборе той или иной программы следует руководствоваться интуицией в оценке класса решающих правил программы и эвристики выбора нужного оператора.

7. ЛИНЕЙНЫЕ ОБУЧАЮЩИЕСЯ МАШИНЫ

Особенностью линейных машин является способ задания решающих правил:

$$gx = \theta \left(\sum_{i=1}^k \alpha_i \varphi_i(x) + \alpha_0 \right), \quad (13)$$

где $\varphi_1(x), \dots, \varphi_k(x)$ — заданная система функций; $\alpha_1, \dots, \alpha_k$ — константы, значения которых определяют конкретное решающее правило.

Считается, что неформальное требование к линейным машинам удовлетворяется выбором системы функций $\varphi_1(x), \dots, \varphi_k(x)$.

Удобно наряду с пространством ситуаций X рассматривать векторное пространство Z , координаты которого $z_i = \varphi_i(x)$ ($i = 1, 2, \dots, k$).

В пространстве Z характеристической функции gx соответствует функция

$$gz = \theta \left(\sum_{i=1}^k \alpha_i z_i + \alpha_0 \right),$$

определяющая, по какую сторону от гиперплоскости $\sum \alpha_i z_i + \alpha_0 = 0$ находится вектор z .

Как уже отмечалось [1, 2], функция роста класса решающих правил (13) ограничена величиной $m^G(l) < l^k + 1$, где k — размерность пространства.

Поэтому согласно теореме 3 для линейных обучающихся машин обеспечена равномерная сходимость эмпирических оценок вероятности ошибок к их истинным значениям по всем решающим правилам класса (13). Следовательно, для каждого выбранного решающего правила (если только длина обучающей последовательности $l \approx Ck \ln k$) истинная оценка вероятности ошибки будет мало отличаться от эмпирической оценки, вычисленной по обучающей последовательности.

Наиболее важным моментом в реализации линейных обучающихся машин является задание алгоритма построения гиперплоскости, разделяющей векторы обучающей последовательности. Алгоритм должен быть таким, чтобы векторы обучающей последовательности, принадлежащие первому классу, по возможности находились по одну сторону от гиперплоскости, а векторы, принадлежащие второму классу, — по другую сторону; число векторов, лежащих в противоположном полупространстве, должно быть минимально.

К сожалению, строго решить задачу о построении такой гиперплоскости удается только в том случае, когда существует гиперплоскость, безошибочно разделяющая векторы обучающей последовательности. Этот случай имеет место при частной постановке задачи обучения

распознаванию образов, т. е. когда известно, что учитель делит ситуации с помощью решающего правила, принадлежащего классу (13), и требуется восстановить это правило в классе (13).

В книге описаны *линейные машины, решающие частную задачу обучения распознаванию образов*. Эти машины отличаются друг от друга способом построения разделяющей гиперплоскости и особенностью построенной гиперплоскости. Одни из них строят разделяющую гиперплоскость с помощью рекуррентных процедур, в то время как другие — нерекуррентным способом.

В случае, когда векторы обучающей последовательности могут быть разделены гиперплоскостью, существует множество разделяющих гиперплоскостей, и каждая из машин строит одну из них.

Рассмотренные выше теоремы не выявляют преимущество одной разделяющей гиперплоскости перед другой. Однако эвристически видимо лучше выбирать такую разделяющую гиперплоскость, которая наиболее далеко отстоит от ближайших к ней векторов обучающей последовательности. Такая гиперплоскость единственна (алгоритмы для построения этой гиперплоскости также описаны в книге). Эти алгоритмы обладают еще одной особенностью: с их помощью устанавливается (в случае, если это так), что не существует гиперплоскости, разделяющей векторы обучающей последовательности.

Особое место среди линейных обучающихся машин, рассмотренных в книге, занимает алгоритм, описанный в статье Фомина, который решает частную задачу обучения распознаванию образов. Класс решающих правил этой машины имеет вид (13). Однако коэффициенты α_i могут принимать только два значения: $+1$ и -1 . Понятно, что емкость класса решающих правил этой машины значительно меньше соответствующей емкости для линейной машины; следовательно, достаточная длина обучающей последовательности должна быть меньше, чем для обычной линейной машины.

В случае, если не существует гиперплоскости, безошибочно разделяющей элементы обучающей последовательности, (т. е. если решается задача обучения распознаванию образов в общей постановке) необходимо найти такую гиперплоскость, на которой число ошибок на элементах обучающей последовательности минимально. К сожалению, не существует регулярных методов по-

строения подобной гиперплоскости, поэтому она строится эвристически.

В книге рассмотрена *линейная обучающаяся машина*, решающая задачу обучения распознаванию образов в общей постановке. Идея состоит в том, что определяется вектор обучающей последовательности, наиболее затрудняющий построение разделяющей гиперплоскости. Он исключается из обучающей последовательности, и оставшиеся векторы вновь разделяются гиперплоскостью. Если разделение все еще невозможно, то исключается еще один вектор, и так до тех пор, пока оставшееся множество векторов не будет разделено.

8. МАШИНЫ С РАСШИРЯЮЩИМСЯ КЛАССОМ РЕШАЮЩИХ ПРАВИЛ

Из рассмотренных выше теорем вытекает, что при фиксированной длине обучающей последовательности чем меньше емкость класса решающих правил машины, тем вероятность правильной классификации с помощью выбранного решающего правила ближе к вычисленной частоте появления событий. Поэтому наиболее вероятно, что из двух решающих правил, делающих одинаковое число ошибок на одной и той же обучающей последовательности, более близким к требуемому окажется то правило, которое выбрано из класса решающих правил меньшей емкости.

Эти соображения и используются при построении обучающихся машин с расширяющимся классом решающих правил.

Первоначально решающее правило выбирается из достаточно узкого класса; если выбранное правило делает на обучающей последовательности сравнительно много ошибок, то класс решающих правил расширяется и нужное правило отыскивается уже в расширенном классе. Однако при фиксированной длине обучающей последовательности расширять классы решающих правил целесообразно только до известного предела (с расширением класса решающих правил вероятность того, что эмпирическая оценка найденного правила близка к вероятностной, уменьшается).

В книге будут также описаны *машины с расширяющимся классом решающих правил*. Идея состоит в том, что сначала составляется решающее правило в классе

$$gx = \Theta \left(\sum_{i=1}^k \alpha_i \varphi_i(x) + \alpha_0 \right).$$

Затем, если найденное правило

$$g^*x = \Theta \left(\sum_{i=1}^k \alpha_i^* \varphi_i(x) + \alpha_0^* \right)$$

делает на обучающей последовательности сравнительно много ошибок (т. е. в исходном классе нет достаточно хорошего решающего правила), то происходит расширение класса решающих правил и решающее правило отыскивается теперь в классе

$$gx = \Theta \left(\sum_{i=1}^k \alpha_i \varphi_i(x) + \alpha_{k+1} g^*x + \alpha_0 \right).$$

Если же и в расширенном классе решающих правил нет нужного, то возможно дальнейшее расширение.

Геометрически это означает, что сначала строится разделяющая гиперплоскость. Если эта гиперплоскость неправильно классифицирует достаточно много векторов обучающей последовательности, то делается попытка разделить векторы обучающей последовательности гиперплоскостью, составленной из двух гиперплоскостей. Если же построенная гиперплоскость также неправильно классифицирует достаточно много векторов обучающей последовательности, то строится гиперповерхность, составленная из трех гиперплоскостей и т. д.

Идея алгоритма — получить достаточно хорошее разделение векторов обучающей последовательности с помощью минимального числа гиперплоскостей (чем меньше гиперплоскостей, тем надежнее работа построенного решающего правила).

Машина, реализующая метод потенциальных функций, выбирает решающее правило из класса

$$gx = \Theta(\sum \alpha_i \varphi_i(x) + \alpha_0).$$

Особенность этой машины состоит в том, что число функций $\varphi_i(x)$, вообще говоря, заранее не фиксировано, и поэтому априори нельзя сказать, сколько функций

$\varphi_i(x)$ с весовым коэффициентом $|\alpha_i| < \delta$ образуют выбранное решающее правило. Число таких функций, вообще говоря, растет с увеличением длины обучающей последовательности; однако надлежащим выбором потенциальной функции можно добиться достаточно медленного роста.

9. КВАЗИЛИНЕЙНЫЕ ОБУЧАЮЩИЕСЯ МАШИНЫ

Так же как и у линейных машин, класс решающих правил квазилинейных машин задается в виде

$$gx = \theta \left(\sum_{i=1}^k \alpha_i \varphi_i(x) + \alpha_0 \right).$$

Отличие состоит в том, что если у линейных машин k — число небольшое (100—300), а параметры α могут принимать любые значения, то у квазилинейных машин k — число большое (порядка 10^5 — 10^6), а параметры α могут принимать лишь ограниченные значения. При этом удачно сконструированные квазилинейные машины имеют примерно такую же емкость класса решающих правил, как и линейные машины. Для таких машин справедливы достаточные условия равномерной сходимости, приведенные выше.

В книге квазилинейные машины представлены двумя типами.

В машине «Кора» в качестве функций $\varphi_i(x)$ выбраны все логические функции от τ переменных ($\tau=2, 3, 4$), заданные на всех подпространствах размерности τ пространства E_n (E_n — пространство бинарных векторов размерности n). Параметры же α могут принимать только три значения: $-1, 0, +1$. Причем из p параметров, отличных от нуля, половина принимают значение 1, а половина -1 .

Для машины «Кора» легко можно оценить число различных на множестве E_n решающих правил.

В самом деле, в пространстве E_n число различных подпространств размерности τ равно C_n^τ (оно оценивается величиной n^τ). Число логических функций от τ переменных в каждом подпространстве равно 2^{2^τ} . Откуда число N_φ функций φ может быть оценено $N_\varphi < 2^{2^\tau} n^\tau$.

Так как только половина функций φ суммируются с коэффициентом $+1$, а половина — с коэффициентом -1 , то число N различных решающих правил оценивается $N < (2^{2^\tau} n^\tau) \cdot p$.

Как уже отмечалось, в случае, когда число различных решающих правил ограничено величиной N , выполняются достаточные

условия равномерной сходимости, при этом достаточная длина обучающей последовательности оценивается как $l \approx C \ln N$. Для этой машины

$$l \approx Cp(\tau \ln n + 2^\tau).$$

Поиск решающего правила, которое делает на обучающей последовательности минимальное число ошибок, ведется эвристически. Особенность машины «Кора» и заключается в использовании некоторых эвристических приемов для построения такого решающего правила.

Идея машины, описанной В. А. Браиловским и А. Л. Лунцем, несколько упрощенно состоит в следующем. Так же как и «Кора», эта машина обучается классификации бинарных векторов.

Функции $\varphi_i(x)$ заданы во всех подпространствах размерности τ ($\tau=2, 3, 4$) пространства E_n . В каждом таком подпространстве задано 2^τ функций (по числу вершин τ -мерного куба), причем j -я функция ставит в соответствие j -й вершине 1, а всем остальным вершинам — 0.

Пусть N_φ — число таких функций (N_φ может быть легко вычислено: $N_\varphi = 2^\tau C_n^\tau$).

Класс решающих правил машины задается в виде

$$gx = \theta \left(\sum_{i=1}^{N_\varphi} \alpha_i \varphi_i(x) + \alpha_0 \right), \quad (14)$$

где коэффициенты α_i могут принимать любые значения, при условии, что отличные от нуля коэффициенты α соответствуют функциям, заданным не более чем в p подпространствах (всего не более $2^\tau p$ величин).

Оценим функцию роста $m^G(l)$ рассмотренного класса решающих правил. Согласно теореме 2 функция $m^G(l)$ либо тождественно равна 2^l , либо $m^G(l) < l^k + 1$, где k — минимальное число векторов, для которых $m^G(k) \neq 2^k$.

Для того чтобы оценить функцию роста, достаточно найти число k . Зафиксируем некоторый набор из p τ -мерных подпространств, т. е. $2^\tau p$ функций, которые в правилах (14) могут быть отличны от нуля.

Согласно [1, 2] с помощью подбора коэффициентов α для фиксированных функций l векторов могут быть разделены на два класса не более чем $l^{2^\tau p}$ способами. Если же набор из p τ -мерных подпространств не фиксирован, то возможно не более $n^{\tau p}$ таких наборов подпространств, в каждом из которых l векторов могут быть разделены на два класса не более чем $l^{2^\tau p}$ способами. Отсюда всего способов меньше $l^{2^\tau p} n^{\tau p}$,

Найдем теперь такое k , при котором

$$l^{2^{\tau}} p n^{\tau p} < 2^l. \quad (15)$$

Неравенство (15) выполнится при любом $l > l^*$:

$$l^* = (2^{\tau} \ln 2^{\tau} p + \tau \ln n) p.$$

Таким образом, с помощью правил (14) всеми возможными способами могут делиться на два класса не более $l=l^*$ векторов, поэтому функция роста класса решающих правил этой машины ограничена величиной $m^g(l) < l^{l^*} + 1$.

Следовательно, достаточная длина обучающей последовательности оценивается как

$$l \approx C (2^{\tau} \ln 2^{\tau} p + \tau \ln n) p \ln [p 2^{\tau} \ln 2^{\tau} p + \tau p \ln n].$$

Особенность этих машин заключается в отыскании решающего правила (14), минимизирующего число ошибок на обучающей последовательности.

10. ПАРАМЕТРИЧЕСКОЕ СЕМЕЙСТВО КВАЗИЛИНЕЙНЫХ ОБУЧАЮЩИХСЯ МАШИН

В реализациях линейных и квазилинейных машин много общего. Это позволяет построить параметрическое семейство машин, содержащее как линейные, так и квазилинейные машины.

Класс решающих правил такой машины задан в виде

$$gx = \Theta \left(\sum_{i=1}^p \alpha_i \Psi_i(x) + \alpha_0 \right), \quad \Psi(x) = \Theta \left(\sum_{j=1}^{\tau} \beta_j x_j + \beta_0 \right), \quad (16)$$

где p и τ — параметры машины; α_i и β_j — константы, значения которых определяют конкретное решающее правило.

Функция $\Psi(x)$ определяет положение вектора x относительно гиперплоскости:

$$\sum_{j=1}^{\tau} \beta_j x_j + \beta_0 = 0.$$

Коэффициенты β_j могут принимать любые значения при условии, что число ненулевых значений не более τ (τ — заданный параметр машины). Коэффициенты α_i могут принимать только три значения: 0, -1 , $+1$, причем t коэффициентов отлично от нуля, из них t_1 имеют значение $+1$, а $t-t_1$ — значение -1 .

При $\tau=n$ (где n — размерность пространства E_n) и $t=t_1=1$ класс решающих правил квазилинейной машины совпадает с классом решающих правил линейной машины. Если же $\tau \ll n$, $p \gg 1$, то класс решающих правил машины совпадает с классом правил квазилинейной машины.

Покажем, что у рассматриваемой машины емкость класса решающих правил ограничена при любых фиксированных значениях τ и p . Действительно, в каждом фиксированном подпространстве размерности τ гиперплоскость разделит l векторов не более чем l^τ способами; в p фиксированных подпространствах p гиперплоскостей разделят l векторов на два класса не более чем $l^{\tau p}$ способами. Если же p подпространств не фиксировано, то количество способов разделения l векторов на два класса не превосходит $l^{\tau p} \cdot n^{\tau p}$.

Максимальное число векторов, которые всеми возможными способами разделяются на два класса решающими правилами (16), может быть найдено из уравнения $l^{\tau p} n^{\tau p} = 2l$.

Корень этого уравнения $l^* \leq \tau p \ln p \tau n$.

Согласно теореме 2 для класса (16) при любых фиксированных значениях τp справедливы достаточные условия равномерной сходимости, и при этом достаточная длина обучающей последовательности оценивается величиной

$$l \approx C \tau p \ln p \tau n \cdot \ln(p \tau \ln p \tau n).$$

Особенность алгоритмов рассмотренного параметрического семейства машин заключается в способе выбора решающего правила (16), делающего минимальное число ошибок на обучающей последовательности.

11. МАШИНЫ, ПРЕДНАЗНАЧЕННЫЕ ДЛЯ РЕШЕНИЯ ЗАДАЧ МЕДИЦИНСКОЙ ДИАГНОСТИКИ

Особое место в применении обучающихся машин занимают задачи медицинской дифференциальной диагностики. Применительно к этим задачам успешно используются многие рассмотренные здесь алгоритмы.

Однако специфика медицинских задач и их важность делают целесообразным создание специализированных обучающихся машин.

Формально специализированные машины сделаны по той же схеме, что и другие машины, описанные в этой книге: выбирается решающее правило, делающее минимальное число ошибок на обучающей последовательности.

Успех в работе специализированных машин определяется более тонкими условиями, чем условия равномерной сходимости, а именно условиями, учитывающими свойства распределения $P(x)$ (т. е. свойствами тех задач, которые предстоит решать с помощью этих алгоритмов).

Свойства распределения $P(x)$ априори неизвестны и поэтому установить эффективность машины для решения определенного класса задач можно только в случае глубоких экспериментальных исследований.

Такие исследования были проведены для решения задач дифференциальной медицинской диагностики с помощью приведенных специализированных машин. Большое количество решенных задач дает основание рекомендовать применение этих машин для решения задач медицинской диагностики.

К задачам обучения распознаванию образов примыкает задача обучения без учителя или, как ее еще называют, «задача получения объективной классификации».

Суть задачи состоит в следующем: пусть различным классам объектов в пространстве параметров объектов соответствуют изолированные, достаточно удаленные друг от друга группы точек. Требуется разбить векторы обучающей последовательности на группы так, чтобы в каждой группе были «близкие» между собой векторы, а в разных группах — векторы, далекие между собой. Эта задача может быть формализована и решена с помощью методов, близких к методам обучения распознаванию образов.

В книге приведены также алгоритмы обучения классификации.

СПИСОК ЛИТЕРАТУРЫ

1. Вапник В. Н., Червоненкис А. Я. Об одном классе алгоритмов обучения распознаванию образов. «Автоматика и телемеханика», т. XXV, 1964, № 6.

2. Нильсон Н. Обучающиеся машины. Изд-во «Мир», 1967.

3. Вапник В. Н., Червоненкис А. Я. О равномерной сходимости частот появления событий к их вероятностям. «Теория вероятностей и ее применения», т. XVI, 1971, вып. 2.

Обучаемые опознающие системы и рекуррентные конечно-сходящиеся алгоритмы

Сформулируем три рекуррентных конечно-сходящихся алгоритма, описывающих работу блока обучения опознающих систем.

1. ОСНОВНЫЕ СИСТЕМЫ НЕРАВЕНСТВ

Предположим, что A и B — некоторые непересекающиеся множества объектов, задаваемых векторами; $s(x)$ — функция, определенная на этих множествах:

$$s(x) = \begin{cases} 1, & \text{если } x \in A, \\ -1, & \text{если } x \in B. \end{cases} \quad (1.1)$$

Задача построения обучаемой опознающей системы сводится к задаче восстановления функции $s(x)$ по заданным ее значениям на некоторых векторах из множеств A и B [1]. Совокупность этих векторов называется обучающим множеством*), которое обозначим через

$\tilde{A} \cup \tilde{B}$ (здесь \tilde{A} — конечное число векторов множества A ; \tilde{B} — конечное число векторов множества B , а $\tilde{A} \cup \tilde{B}$ — их объединение). В некоторых интересных задачах множества A и B имеют весьма сложную структуру.

В связи с этим обычно конструируется «спрямляющее» пространство, т. е. на множестве векторов исходя из некоторых соображений задаются вещественные функции $a_j(x)$, $j=1, 2, \dots, N$ (значение $a_j(x)$ является реакцией некоторого « a -элемента» на вектор x). Вместо множеств A и B рассматриваются множества A' и B' в N -мерном евклидовом пространстве R_N , полученные при отображении A и B с помощью заданных a -элементов, т. е. при отображении $y_j = a_j(x)$, $j=1, 2, \dots, N$; $y_j \in R_N$.

Во многих случаях множества A' , B' могут быть раз-

*) Если элементы обучающего множества перенумерованы, то их называют обучающей последовательностью.

делены плоскостью (отсюда и произошел термин «спрямляющее» пространство).

После того как a -элементы $a_j(x)$, $j=1, 2, \dots, N$ выбраны, в качестве приближения к функции $s(x)$ можно взять функцию $\text{sign } s_N(x)$:

$$s_N(x) = \sum_{j=1}^N \kappa_j a_j(x), \quad (1.2)$$

где κ_j — числовые коэффициенты, подлежащие определению.

Набор этих коэффициентов всегда определяет некоторую плоскость в спрямляющем пространстве, и если функции $a_j(x)$ выбраны удачно, то

$$s(x) = \text{sign } s_N(x), \quad x \in A \cup B. \quad (1.3)$$

Искомые значения κ_j , таким образом, должны быть решениями системы неравенств

$$\sum_{j=1}^N \kappa_j a_j(x) s(x) > 0, \quad x \in A \cup B. \quad (1.4)$$

Поскольку известна лишь конечная обучающаяся последовательность, то естественно находить значение κ_j из решения системы неравенств

$$\sum_{j=1}^N \kappa_j a_j(x) s(x) \geq \varepsilon, \quad x \in \tilde{A} \cup \tilde{B}. \quad (1.5)$$

Здесь ε — некоторое неотрицательное число. (Относительно более точного обоснования перехода от системы (1.5) к системе (1.4), см., например, [1].)

Система неравенств (1.5) является первой системой неравенств, решению которых посвящена настоящая работа. Вместо системы (1.5) иногда удобнее рассмотреть систему

$$\sum_{j=1}^N \kappa_j a_j(x) s(x) \geq \varepsilon \sqrt{\sum_{j=1}^N \kappa_j^2} \sqrt{\sum_{j=1}^N a_j^2(x)}, \quad x \in \tilde{A} \cup \tilde{B}. \quad (1.6)$$

Действительно, при умножении неравенств (1.5) на одно и то же число γ получится аналогичная система, в которой $\kappa'_j = \gamma \kappa_j$, $\varepsilon' = \gamma \varepsilon$.

Поэтому неравенства (1.5) имеет смысл решать либо при условии ограниченности вектора u , либо при более сильном дополнительном условии нормирования вектора:

$$u = \|\kappa_1, \dots, \kappa_N\|. \quad (1.7)$$

В последнем случае система (1.5) принимает вид (1.6).

Система (1.6) является второй системой, решению которой посвящена настоящая работа.

Для значений κ_j , удовлетворяющих неравенствам (1.5), (1.6), функция (1.2) осуществляет аппроксимацию «по знаку» искомой функции $s(x)$. Имеет смысл также рассмотреть равномерную аппроксимацию, т. е. аппроксимацию вида

$$\left| s(x) - \sum_{j=1}^N \kappa_j a_j(x) \right| < \varepsilon, \quad x \in A \cup B, \quad (1.8)$$

где $\varepsilon > 0$ — заданное число, определяющее степень приближения.

Поскольку заданы лишь векторы обучающей последовательности $\tilde{A} \cup \tilde{B}$, то вместо системы (1.8) получаем конечную систему неравенств

$$\left| s(x) - \sum_{j=1}^N \kappa_j a_j(x) \right| < \varepsilon, \quad x \in \tilde{A} \cup \tilde{B}. \quad (1.9)$$

Система неравенств (1.9) является третьей системой, решение которой будет рассмотрено ниже.

Многие задачи, близкие к задачам распознавания образов, например задачи восстановления характеристик $s(x)$ вектора по данным наблюдения, также сводятся к решению неравенств (1.9). В этих случаях $s(x)$ — произвольная функция, отличная, вообще говоря, от функции вида (1.1). Поэтому в дальнейшем будет рассмотрен случай, когда в системе (1.9) $s(x)$ — произвольная функция.

Для сокращения записи удобно ввести следующие векторные обозначения. Обозначим через u искомый вектор с компонентами κ_j , $j=1, 2, \dots, N$: $u = \|\kappa_1, \dots, \kappa_N\|$, через u_k — некоторое k -е приближение к вектору u : $u_k = \|\kappa_1^{(k)}, \dots, \kappa_N^{(k)}\|$.

Наконец, через $c(x)$ обозначим вектор-функцию

$$c(x) = \|s(x) a_1(x), s(x) a_2(x), \dots, s(x) a_N(x)\|.$$

Напомним, что на векторах обучающего множества $\tilde{A} \cup B$ значения вектор-функции $c(x)$ известны. Через c_k обозначим вектор

$$c_k = c(x_k), \quad x_k \in \tilde{A} \cup \tilde{B}. \quad (1.10)$$

В этих обозначениях системы (1.5), (1.6) и (1.9) можно записать соответственно в виде

$$(u, c_k) + \gamma_k > 0, \quad (1.11) \quad (u, c_k) > \varepsilon \|u\| \cdot \|c_k\|; \quad (1.12)$$

$$|(u, c_k) + \gamma_k| < \varepsilon, \quad (1.13)$$

где $k = 1, 2, \dots, m$ (m — длина обучающей последовательности); ε — положительное число; γ — произвольное

число; c_k — произвольные векторы, $\|u\| = \sqrt{\sum_{j=1}^N x_j^2}$.

Ниже будут рассматриваться именно системы неравенств (1.11) — (1.13).

2. ЦИКЛИЧНО-СХОДЯЩИЕСЯ И КОНЕЧНО-СХОДЯЩИЕСЯ АЛГОРИТМЫ

Предполагая, что x_1, \dots, x_m — некоторая заранее заданная конечная последовательность векторов, продолжим ее периодически, считая, что $x_{k+m} = x_k$ для произвольного натурального k . Полагая $c_{k+m} = c_k$, $\gamma_{k+m} = \gamma_k$, будем считать, что каждая из систем неравенств (1.11) — (1.13) также периодически продолжена. Такие системы бесконечных неравенств (1.11) — (1.13) назовем *циклическими*.

Далее нас будут интересовать лишь рекуррентные алгоритмы решения систем (1.11) — (1.13), т. е. алгоритмы вида $u_{k+1} = g(u_k, c_k, \gamma_k)$, где $k = 1, 2, \dots$. Здесь u_1 — произвольный начальный вектор, а g — некоторая вектор-функция своих аргументов.

В ряде случаев удастся построить лишь алгоритм несколько более сложного вида, чем (2.1), а именно:

$$w_{k+1} = g(w_k, c_k, \gamma_k), \quad k = 1, 2, \dots, \quad (2.1a)$$

где $w_k = \begin{pmatrix} u_k \\ v_k \end{pmatrix}$; v_k — вектор некоторых дополнительных параметров (при этом необходимо указать вектор v_1).

Алгоритм вида (2.1а) также является рекуррентным. Естественно, что размерность вектора v желательно сделать возможно меньшей.

Алгоритм вида (2.1) или (2.1а) называется циклично сходящимся, если начиная с некоторого $k=k_0$ окажется, что $u_{k_0}=u_{k_0+1}=\dots=u_{k_0+m-1}$ и если для значений $k=k_0, k_0+1, \dots, k_0+m-1$ выполнены соответствующие неравенства (1.11)—(1.13). Очевидно, что значение u_k будет при этом являться решением исходной системы m неравенств.

Системы неравенств удобно решать с помощью рекуррентных процедур. (Алгоритмы рекуррентного типа дают возможность не запоминать входную информацию, объем которой обычно весьма велик, или запоминать ее вне блока обучения.)

Пусть работа блока обучения машины описывается уравнением (2.1), где c_k, γ_k — входные данные, определяемые по входному вектору x_k и по соответствующему «правильному» ответу $s(x_k)$, сообщаемому учителем. На фиксированном k -м шаге обучения ответом машины является число $\text{sign } s_N(x_k)$, где $s_N(x_k)$ определяется формулой (1.2), в которой вектором параметров $u=u_k$ служит вектор, вычисленный на этом шаге блоком обучения.

Естественно, что ответ машины, вообще говоря, не совпадает с правильным ответом $s(x_k)$. Будем считать (и в этом состоит особенность рассматриваемого случая), что вектор x_{k+1} , подаваемый для обучения на следующем шаге, может зависеть от предыдущего ответа машины, а также, возможно, и от всех ранее данных ответов машины. (Если машина ошиблась на некотором векторе x_k , естественно его еще раз предъявить, т. е. $x_{k+1}=x_k$).

Таким образом, будем считать, что вектор x_k выбирается в зависимости от предыдущих ответов машины, т. е. в зависимости от последовательности u_1, u_2, \dots, u_k , а также от ранее предъявленных ей векторов и от номера k :

$$x_k = X(k, x_1, \dots, x_{k-1}, u_1, \dots, u_k), \quad (2.2)$$

причем X — некоторая детерминированная или случайная функция своих аргументов.

В формуле (2.2) учтены все величины, от которых может зависеть x_k ; более того, можно сказать, что эта формула определяет как процесс обучения распознава-

нию образов, так и процесс распознавания после обучения. Действительно, любое фиксированное правило, согласно которому на m -м шаге происходит остановка обучения, означает, что при $k \geq m$ векторы предъявляются по некоторому новому закону — закону «экзамен», который также описывается формулой (2.2).

Подчеркнем, что имеется игровая ситуация: k -й вектор x_k на входе машины зависит от ее предыдущих ответов, а $(k+1)$ -й вектор параметров u_{k+1} зависит от предъявленного вектора x_k . Для произвольного u_1 согласно выражению (2.2) получим $x_1 = X(1, u_1)$; далее по x_1 и $s(x_1)$ определим c_1, γ_1 . Затем по формулам (2.1), (2.2) найдем $u_2, x_2 = X(2, x_1, u_1, u_2), u_3, x_3, x_4, u_4, \dots$

В рассматриваемой ситуации неравенства (1.11), (1.13) заранее не заданы, — они появляются на каждом шаге в зависимости от предыдущих значений u_1, \dots, u_{k-1} .

Определение. Алгоритм (2.1) называется *конечно-сходящимся алгоритмом* решения неравенств (1.11), (1.12) или (1.13), если для любого u_1 и для любого алгоритма типа (2.2) найдется номер L , такой, что при $k \geq L$ будут выполнены равенства

$$u_L = u_{L+1} = \dots \quad (2.3)$$

и все неравенства (1.11) или (1.12), или (1.13), в которых $u = u_L$ и $k \geq L$.

Таким образом, конечно-сходящийся алгоритм через конечное число шагов дает решение бесконечного числа неравенств.

Отметим, что для определения параметра u_{k+1} необходимо знать не только вектор x_k , но и значение $s(x_k)$. Однако при $k \geq L$ в силу (2.3) знание $s(x_k)$ уже не требуется. Поэтому если длина m обучающей последовательности удовлетворяет неравенству $m \geq L$, то машина может работать в режиме «экзамен» ($k > m$) с фиксированным $u = u_L$ и будет при этом опознавать верно каждый вектор. С другой стороны, число L зависит от алгоритма (2.2), поэтому на любом фиксированном шаге (даже когда равенство $u_m = u_{m+1} = \dots = u_{m+p}$ выполняется многократно и удовлетворяются все соответствующие неравенства) нельзя установить, имеет ли место неравенство $m \geq L$ или нет.

Избавиться от этого недостатка без использования каких-либо дополнительных сведений о «закономерности» появления векторов x_k принципиально невозможно.

Действительно, априори, без привлечения каких-либо сведений о распределении векторов x на множествах A и B невозможно сказать, достаточно ли «представительна» обучающая последовательность, а следовательно, невозможно «увязать» задаваемое нами число m с числом L . Однако при достаточно общих предположениях любой конечно сходящийся алгоритм вычисляет «хорошее» значение u_m , т. е. вероятность неправильного ответа на «экзамене» может быть сколь угодно малой, если число m достаточно велико [1] (предполагается, что векторы x_k для $k=1, 2, \dots, m$ предъясняются случайно в соответствии с некоторым, вообще говоря, неизвестным, распределением на множестве $A \cup B$).

Отметим, что всякий конечно-сходящийся алгоритм является, очевидно, и циклично-сходящимся. Поэтому каждый конечно-сходящийся алгоритм можно применить для решения конечной системы неравенств (1.5), (1.6) и (1.9). Условие останова при этом определяется требованием выполнения соотношений (2.3) и соответствующих неравенств для $k=k_0, k_0+1, \dots, k_0+m-1$.

3. ОПИСАНИЕ АЛГОРИТМОВ

Рассмотрим рекуррентные конечно-сходящиеся алгоритмы нахождения решений систем неравенств (1.11), (1.12) и (1.13).

Предположим, что векторы c_k могут принимать любые значения из некоторого ограниченного множества C :

$$\frac{\sup}{c \in C} |c| < T,$$

где T — положительная постоянная, а множество C состоит из всех векторов вида $c(x)$, x пробегает множество $A \cup B$.

Рассмотрим неоднородную систему

$$(u, c_k) > \varepsilon, \quad k=1, 2, \dots$$

с добавочным ограничением, состоящим в том, что как векторы c_k , так и искомый вектор u должны находиться на единичной гиперсфере. Последнее обстоятельство позволяет дать простую геометрическую интерпретацию решениям системы (1.12). Если предположить, что множества A' и B' спроектированы на единичную гипер-

сферу, то всякое решение u системы (1.12) определяет плоскость, разделяющую множества A' и B' и проходящую от этих множеств не ближе чем на величину ε .

Предположим, что неравенства (1.11)—(1.13) разрешимы и допустим, что существует такой вектор u_* и такое положительное число ε_* [свой для каждой из систем (1.11)—(1.13)], при которых справедливы неравенства

$$(u_*, c) + \gamma \geq \varepsilon_* > 0, \quad c \in C, \quad \gamma \in \Gamma, \quad (3.1)$$

$$(u_*, c) \geq \varepsilon_* \|c\| \cdot \|u_*\|, \quad \varepsilon_* > \varepsilon, \quad c \in C, \quad \gamma \in \Gamma, \quad (3.2)$$

$$|(u_*, c) + \gamma| \leq \varepsilon_* < \varepsilon, \quad c \in C, \quad \gamma \in \Gamma. \quad (3.3)$$

Разумеется, вектор u_* и число ε_* неизвестны.

Перейдем непосредственно к описанию алгоритмов для нахождения решений систем неравенств (1.11), (1.12) и (1.13). Эти алгоритмы названы соответственно «КС-1», «Отражение» и «Полоска». Во всех этих алгоритмах фигурируют неотрицательные числа $\rho_j, \beta_j, j=1, 2, \dots$, удовлетворяющие условиям

$$0 \leq \beta_j \leq 2; \quad \rho_j \xrightarrow{j \rightarrow \infty} 0, \quad \rho_1 + \rho_2 + \dots = \infty. \quad (3.4)$$

Программы, реализующие эти алгоритмы, унифицированы так, что структура и взаимодействие отдельных блоков для каждой из программ аналогичны; основное отличие состоит в различных переменных программы. Поэтому наиболее подробно даются пояснения к блок-схеме алгоритма «КС-1», в то время как в остальных случаях приводятся лишь краткие пояснения.

Алгоритм «КС-1»

Алгоритм предназначен для нахождения решения u системы неравенств (1.11). Ниже приводится описание алгоритма

1. Параметры программы

N — размерность пространства, в котором решаются неравенства;

$\beta_j, \rho_j, j=1, 2, \dots$ — неотрицательные числа, удовлетворяющие условиям (3.4);

M — параметр останова алгоритма (натуральное число).

2. Переменные программы

$c_h, h=1, 2, \dots$ — векторы размерности N ;
 $\gamma_h, h=1, 2, \dots$ — действительные числа
 (c_h и γ_h — коэффициенты неравенств (1.11)),
 u_h — итерация номера R решения системы неравенств,
 u_1 задается произвольно),

$$u_{h+1} = \begin{cases} u_h, & \text{если } \eta_h > 0, \\ u_h + c_h \zeta_h, & \text{если } \eta_h \leq 0, \quad h = 1, 2, \dots, \end{cases}$$

где $\eta_h = (u_h, c_h) + \gamma_h$;

$j(h)$ — счетчик числа изменений вектора u_h за h шагов алгоритма, $j(1) = 0$,

$$j(h+1) = \begin{cases} j(h), & \text{если } \eta(h) > 0, \\ j(h) + 1, & \text{если } \eta(h) \leq 0, \quad h = 1, 2, \dots; \end{cases}$$

ζ_h — величина, определяемая формулой $\zeta_h = \rho_{j(h)} - \beta_h \eta_h / (c_h, c_h)$;

σ_h — число выполненных неравенств (1.11),

$$\sigma_h = \begin{cases} 0, & \text{если } \eta_h \leq 0, \\ \sigma_{h-1} + 1, & \text{если } \eta_h > 0. \end{cases}$$

3. Описание программы

Блок-схема программы «КС-1» изображена на рис. 1.

0. Блок запуска алгоритма (БЗ)

При получении сигнала о запуске блок подает сигнал $h=1$ на вход блоков I и VIII. Кроме того, в блок II засылается начальный вектор u_1 и управление передается блоку I.

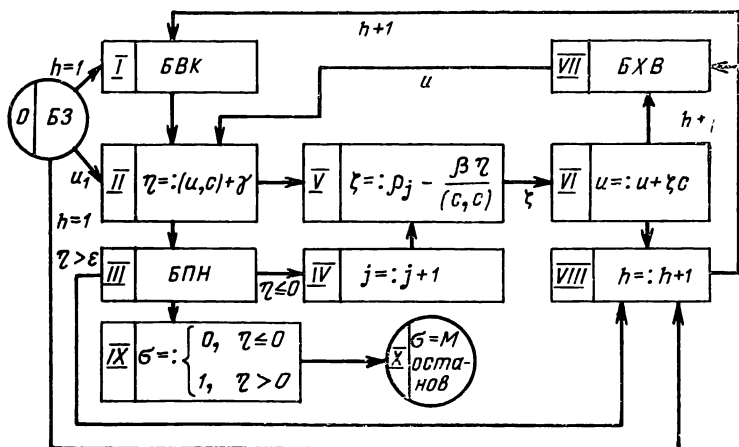


Рис. 1.

1. Блок выдачи коэффициентов неравенств (БВК)

Под действием сигнала h на выходе блока образуется вектор c_h и число γ_h и управление передается блоку II.

II. Блок вычисления промежуточных значений

По векторам c_h , u_h и числу γ_h вычисляются величины (u_h, c_h) , (c_h, c_h) , $\eta_h = (u_h, c_h) + \gamma_h$, $\text{sign } \eta_h$, которые передаются блоку V. Величина η_h поступает также на блок III, которому передается управление.

III. Блок проверки неравенства (БПН)

В блоке происходит проверка очередного неравенства. Если неравенство выполнено, т. е. если $\eta_h > 0$, то в счетчик блока IX засылается единица и управление передается блоку VIII. Если же $\eta_h \leq 0$, то происходит засылка нуля в счетчик блока IX и управление передается блоку IV.

IV. Блок счета числа изменений вектора u_h за h шагов алгоритма

Определяется величина

$$j(h+1) = \begin{cases} j(h), & \text{если } \eta_h > 0, \\ j(h) + 1, & \text{если } \eta_h \leq 0, \end{cases} \quad j(1) = 0.$$

Число $\rho_{j(h)}$ засылается в блок V, а на освободившееся место записывается число $\rho_{j(h+1)}$ и управление передается блоку V.

V. Блок вычисления величины ζ_h

Происходит вычисление величины

$$\zeta_h = \rho_{j(h)} - \beta_h \eta_h / (c_h, c_h),$$

которая засылается в блок VI. Управление передается блоку VI.

VI. Блок изменения вектора u_h

Происходит вычисление вектора

$$u_{h+1} = u_h + \zeta_h c_h,$$

который засылается в блок VII. Управление передается блоку VIII.

VII. Блок хранения вектора u_h (БХВ)

Под действием сигнала $h+1$ вектор u_{h+1} поступает на вход блока II.

VIII. Блок счета числа неравенств

При каждом обращении к этому блоку содержание его счетчика увеличивается на единицу. Результат — число $(h+1)$ засылается в блоки VII и I. Управление передается блоку I.

IX. Блок счета числа выполненных неравенств

В счетчике подсчитывается число выполненных неравенств после последнего изменения вектора u_h ; результат поступает на блок X.

Х. Блок останова работы алгоритма

Если содержимое σ счетчика блока IX окажется равным числу M , то алгоритм заканчивает свою работу. На печать выдаются вектор из блока VII и содержимое счетчика блока IV.

Алгоритм «Отражение»

Алгоритм предназначен для нахождения решения u системы неравенств (1.12). Ниже приводится описание алгоритма.

1. Параметры программы

N — размерность пространства, в котором решаются неравенства;

ε — параметр качества решения, $|\varepsilon| < 1$;

$\beta_j, \rho_j, j=1, 2, \dots$ — неотрицательные числа, удовлетворяющие условиям (3.4);

M — параметр останова алгоритма (натуральное число).

2. Переменные программы

$c_h, h=1, 2, \dots, N$ — векторы коэффициентов неравенств; u_h — итерация решения системы неравенств (1.12).

$$u_{h+1} = \begin{cases} u_h, & \text{если } \eta_h \geq \varepsilon; \\ u_h + \zeta_n (\tilde{c}_h - \varepsilon \tilde{u}_h), & \text{если } \eta_h < \varepsilon, h=1, 2, \dots; \end{cases}$$

$$\tilde{u}_h = \frac{u_h}{(u_h, u_h)^{1/2}}; \quad \tilde{c}_h = \frac{c_h}{(c_h, c_h)^{1/2}}; \quad u_1 \text{ задается произвольно,}$$

η_h — скалярное произведение нормированных векторов \tilde{c}_h и \tilde{u}_h , $\eta_h = (\tilde{c}_h, \tilde{u}_h)$;

$j(h)$ — счетчик числа изменений вектора u_h за h шагов алгоритма, $j(1)=0$,

$$j(h+1) = \begin{cases} j(h), & \text{если } \eta_h \geq \varepsilon, \\ j(h) + 1, & \text{если } \eta_h < \varepsilon, h=1, 2, \dots; \end{cases}$$

$$\zeta_h = \rho_{j(h)} - \beta_h [(u_h c_h) - \varepsilon \|u_h\|] / [1 + \varepsilon^2 - 2\varepsilon (\tilde{c}_h, u_h)];$$

σ_h — число выполненных неравенств после очередного изменения вектора u_h ,

$$\sigma_h = \begin{cases} 0, & \text{если } \eta_h \geq \varepsilon, \\ \sigma_{h-1} + 1, & \text{если } \eta_h < \varepsilon. \end{cases}$$

3. Описание программы

Блок-схема программы «Отражение» изображена на рис. 2. Она отличается от предыдущей лишь блоками IV, V и VI где вычисление величин η_h, ζ_h и u_h происходит по формулам, описанным в § 2. Кроме того, в блоке III происходит сравнение величин η_h и ε ; управление передается блоку IV (если $\eta_h < \varepsilon$) и блоку VIII (если $\eta_h \geq \varepsilon$). В остальном назначение и работа блоков полностью совпадает с описанием, данным для программы «КС-1».

Алгоритм «Полоска»

Алгоритм предназначен для нахождения решения u системы неравенств (1.13).

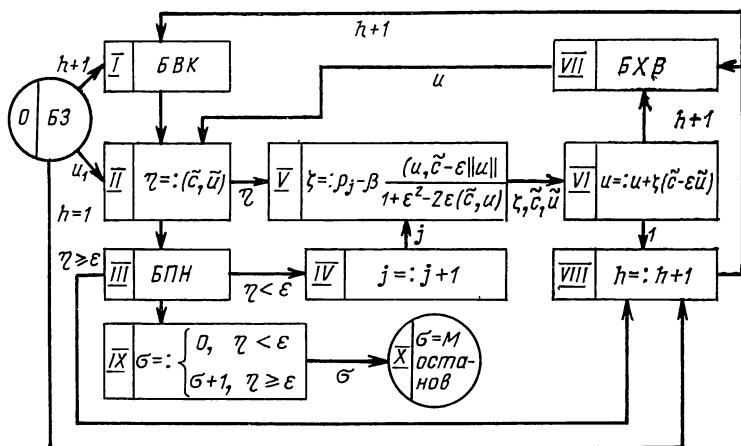


Рис. 2.

1. Параметры программы

N — размерность пространства, в котором решаются неравенства;

ϵ — параметр близости (положительное число);

$\beta_j, \rho_j, j=1, 2, \dots$ — неотрицательные числа, удовлетворяющие условиям (3.4);

M — параметр останова работы алгоритма (натуральное число).

2. Переменные программы

$c_h, h=1, 2, \dots, N$ — векторы коэффициентов неравенств; $\gamma_h, h=1, 2, \dots$ — действительные числа; u_h — итерация номера h решения системы неравенств;

$$u_{h+1} = \begin{cases} u_h, & \text{если } |\eta_h| < \epsilon, \\ u_h + \zeta_h c_h, & \text{если } |\eta_h| \geq \delta; \end{cases}$$

u_1 задается произвольно; η_h — величина, определяемая формулой

$$\eta_h = (u_h, c_h) + \gamma_h;$$

$j(h)$ — счетчик числа изменения вектора u_h за h шагов алгоритма, $j(1)=0$,

$$j(h+1) = \begin{cases} j(h), & \text{если } |\eta_h| < \epsilon, \\ j(h) + 1, & \text{если } |\eta_h| \geq \epsilon, \end{cases}$$

ζ_h — величина, определяемая формулой

$$\zeta_h = -\rho_{j(h)} \operatorname{sign} \eta_h - \beta_h (\eta_h - \epsilon \operatorname{sign} \eta_h) / (c_h, c_h);$$

σ — число выполненных неравенств после очередного изменения вектора u_h ,

$$\sigma_h = \begin{cases} 0, & \text{если } |\eta_h| \geq \varepsilon, \\ \sigma_{h-1} + 1, & \text{если } |\eta_h| < \varepsilon. \end{cases}$$

3. Описание программы

Блок-схема алгоритма «Полоска» изображена на рис. 3. Она отличается от блок-схемы, изображенной на рис. 1, лишь блоком V (где иначе подсчитывается величина ξ_h) и блоком III, где происхо-

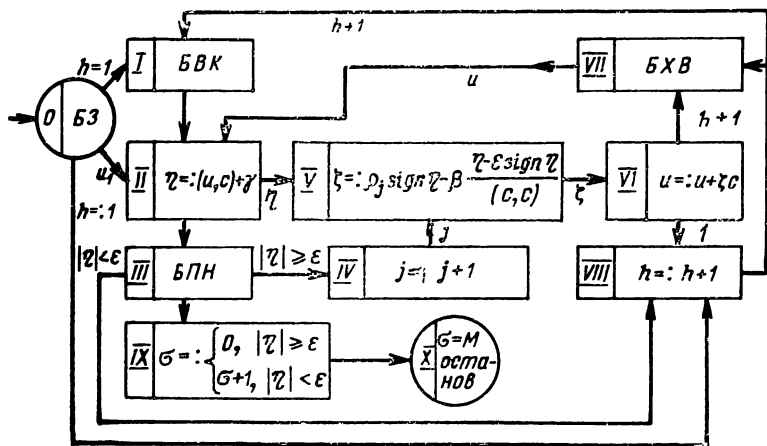


Рис. 3.

дит сравнение величины $|\eta_n|$ с ε . При этом управление передается блоку IV, если $|\eta_n| \geq \varepsilon$, и блоку VIII, если $|\eta_n| < \varepsilon$. В остальном назначение и работа блоков полностью совпадает с описанием, данным для алгоритма «КС-1».

Замечания

1. Алгоритмы «КС-1», «Отражение» и «Полоска» являются конечно сходящимися в смысле данного ранее определения. Конечная сходимость обеспечивается в них наличием слагаемого $\rho_{j(h)}$ в величинах ζ_h , хотя скорость сходимости определяется не этим слагаемым. Если $\rho_{j(h)} \equiv 0$, то конечной сходимости, вообще говоря, нет. Однако если рассматриваемые системы неравенств (1.11) — (1.13) — циклические (см. § 2), то полученным таким образом алгоритмы за конечное число шагов сойдутся к некоторому решению соответствующей системы, т. е. алгоритмы будут циклично сходящимися *).

*) Разумеется, это утверждение подразумевает, что разделяющая гиперплоскость существует.

Если при этом $\beta_h \equiv 2$, то алгоритм «Отражение» существенно упрощается, так как автоматически обеспечиваются равенства $\|u_1\| = \|u_2\| = \dots$. При $\varepsilon = 0$ алгоритм «Отражение» совпадает с алгоритмом «КС-1» при $\gamma_h \equiv 0$. Если в системе неравенств (1.11) $\gamma_h \equiv 0$, т. е. система однородна, то от условия убывания чисел ρ_j при $j \rightarrow \infty$ в алгоритме «КС-1» можно отказаться. В этом случае, в частности, можно взять $\rho_j \equiv \text{const} > 0$.

4. НЕКОТОРЫЕ РЕЗУЛЬТАТЫ МОДЕЛИРОВАНИЯ АЛГОРИТМОВ «КС-1», «ОТРАЖЕНИЕ» И «ПОЛОСКА» НА ЭВМ

1. Отдельные варианты описанных выше алгоритмов были опробованы на ЭВМ для решения ряда практических задач распознавания образов. В одной из таких задач находилась гиперплоскость,

Зависимость скорости обучения от алгоритма обучения (156 неравенств в 154-мерном пространстве)

| Номер эксперимента | Алгоритм обучения | Число циклов обучения | Время обучения, м | Замечания |
|--------------------|---|-----------------------|-------------------|--|
| 1 | „КС-1“ при $\beta_j \equiv 1$, $\rho_j \equiv 0$, $u_1 = c_1$ | 21 | 8 | Найдено решение системы неравенств (1.11) |
| 2 | „КС-1“ при $\beta_j \equiv 2$, $\rho_j \equiv 0$, $u_1 = c_1$ | 10 | 4 | То же |
| 3 | „Отражение“ при $\beta_j \equiv 2$, $\rho_j \equiv 0$, $\varepsilon = 0$, $u_1 = (1, 1, \dots, 1)$ | 10 | 40 | То же |
| 4 | „Отражение“ при $\beta_j \equiv 2$; $\beta_j \equiv 0$; $\varepsilon = 1/16$, $u_1 = u_{\text{вых экспз}}$ | 25 | 40 | Найденное решение удовлетворяет 79% неравенств системы (1.12) и всем неравенствам системы (1.11) |
| 5 | „Отражение“ при $\beta_j \equiv 2$, $\rho_j \equiv 0$; $\varepsilon = 1/32$, $u_1 = u_{\text{вых эксп4}}$ | 25 | 40 | Найденное решение удовлетворяет 97% неравенств системы (1.12) и всем неравенствам системы (1.11) |
| 6 | „Полоска“ при $\rho_j \equiv 0$; $\varepsilon = 0,9$; $u_1 = c_1$ | 37 | 12 | Найдено решение системы неравенств (1.13). |

Примечание: Для всех неравенств $\gamma_h \equiv 0$.

разделяющая два множества в 154-мерном спрямляющем пространстве. Каждое множество было представлено 78 векторами *).

Эксперименты на машине М-220 показали, что скорость обучения существенно зависит от порядка предъявления векторов. Обычно лучший результат достигается при предъявлении представителей различных классов попеременно. После показа всей обучающей последовательности она в том же порядке вновь предъявлялась машине до тех пор, пока машина не обучалась полностью (т. е. пока не находилось нужное решение системы неравенств). Число показов всей обучающей последовательности записывается числом циклов обучения. Эта величина характеризует скорость обучения — важнейшее свойство любого алгоритма.

2. Некоторые данные, характеризующие свойства отдельных вариантов, описанных в § 3 алгоритмов, приведены в таблице.

Достоинством приведенных алгоритмов является то, что при увеличении длины обучающей последовательности время обучения увеличивается не очень быстро, так как число циклов обучения остается примерно тем же.

Наибольшее число неравенств, решение которых было найдено с помощью рекуррентных алгоритмов, составляло 400. Этот предел, однако, связан не с возможностями алгоритмов, а с возможностью получения необходимого экспериментального материала. Решение четырехсот неравенств в 154-мерном пространстве было найдено с помощью алгоритма «КС-1» при $\rho_j = 0$, $\beta_j = 0$ за 8 циклов (10 минут работы машины М-220), что свидетельствует об эффективности рекуррентных алгоритмов, особенно для задач с большой размерностью.

СПИСОК ЛИТЕРАТУРЫ

1. Я к у б о в и ч В. А. Некоторые общие теоретические принципы построения обучаемых опознающих систем. В сб. «Вычислительная техника и вопросы программирования». Изд. ВЦ ЛГУ, 1965, вып. 4.

Б. Н. Козинец

Рекуррентный алгоритм разделения выпуклых оболочек двух множеств

Задача распознавания образов часто сводится к разделению выпуклых оболочек двух множеств в спрямляющем пространстве. Ниже рассмотрен рекуррентный алгоритм разделения гиперплоскостью выпуклых оболо-

*) Рассматривалась задача моделирования опознающей системы, способной обучаться различать рукописные цифры. В спрямляющем пространстве множества были расположены в вершинах единичного куба.

чек двух заданных множеств. В случае, если выпуклые оболочки заданных множеств пересекаются, алгоритм дает возможность обнаружить, что искомой разделяющей плоскости не существует.

ПОСТАНОВКА ЗАДАЧИ

1. В n -мерном евклидовом пространстве заданы два множества точек (векторов) $M = \{x^k\}_{k=1}^m$ и $N = \{y^l\}_{l=1}^p$ *).

Требуется определить, существует ли гиперплоскость, разделяющая эти множества, и если такая гиперплоскость существует, то найти ее.

2. Если разделяющие плоскости существуют, то желательно не только найти одну из них, но и решить более сложную задачу: среди всех разделяющих плоскостей выбрать такую, расстояние от которой до множества $M \cup N$ максимально **). Другими словами, искомая гиперплоскость должна разделять множества M и N и, кроме того, должна быть удалена от этих множеств на возможно большее расстояние. Обозначим через \bar{M} и \bar{N} выпуклые оболочки множеств M и N . Нетрудно доказать, что искомая гиперплоскость проходит через середину отрезка, соединяющего две ближайшие точки множеств \bar{M} и \bar{N} перпендикулярно к нему. Следовательно, задача п. 1 с условием п. 2 сводится к следующей: найти две ближайшие точки множеств \bar{M} и \bar{N} . Если при этом окажется, что расстояние между найденными точками равно нулю, то разделяющей гиперплоскости не существует.

*) Через x^k обозначены радиусы-векторы точек n -мерного пространства с координатами $x_1^k, x_2^k, \dots, x_n^k$. В дальнейшем под „точкой“ понимается ее радиус-вектор. Запись $(x-z, y-z)$ обозначает скалярное произведение вектора $(x-z)$ на вектор $(y-z)$ и вычисляется по формуле

$$(x-z, y-z) = \sum_{i=1}^n (x_i - z_i)(y_i - z_i) = |x-z| |y-z| \cos \alpha,$$

где α — угол между векторами $(x-z)$ и $(y-z)$, а x_i, y_i, z_i ($i=1, 2, \dots, n$) — соответственно координаты векторов x, y, z .

**) Расстояние от разделяющей плоскости P до множества $M \cup N$ определяется по формуле $d = \min_{z \in M \cup N} \rho(z, P)$, где $\rho(z, P)$ — расстояние от точки z до плоскости P , равное длине перпендикуляра, опущенного из точки z на плоскость P .

3. При этом для решения задачи, поставленной в пп. 1 и 2. желательно использовать такой алгоритм, реализация которого требовала бы возможно меньшей памяти. Алгоритм, рассмотренный ниже, удовлетворяет этому требованию; он «должен помнить» на каждом шаге информацию в количестве, необходимом для запоминания координат двух векторов размерности n . Однако этот алгоритм обладает тем недостатком, что существуют множества M и N , разделение которых осуществляется лишь за бесконечное число шагов. В действительности для задач распознавания образов не требуется точно решать задачу, поставленную в п. 2; достаточно найти пару точек из множеств \bar{M} и \bar{N} , расстояние между которыми близко к расстоянию между множествами \bar{M} и \bar{N} .

Эту задачу можно сформулировать следующим образом: *по заданному числу ε ($0 \leq \varepsilon \leq 1/2$) требуется найти такие две точки $x \in \bar{M}$ и $y \in \bar{N}$, для которых выполняются условия:*

$$\left. \begin{aligned} (x^i - x, y - x) &\leq \varepsilon (y - x, y - x) && \text{для всех } x^i \in M, \\ (y^i - y, x - y) &\leq \varepsilon (x - y, x - y) && \text{для всех } y^i \in N. \end{aligned} \right\} \quad (1)$$

Если при этом расстояние между точками x и y будет равно нулю, то это означает, что разделяющей плоскости между множествами M и N не существует.

Выясним геометрический смысл условия (1). Пусть точки $x \in \bar{M}$ и $y \in \bar{N}$ удовлетворяют условию (1). Условие (1) можно переписать следующим образом:

$$\left. \begin{aligned} |x^i - x| \cos \alpha_i &\leq \varepsilon |y - x| && \text{для всех } x^i \in M, \\ |y^i - y| \cos \beta_i &\leq \varepsilon |y - x| && \text{для всех } y^i \in N, \end{aligned} \right\} \quad (2)$$

где α_i — угол между векторами $(x^i - x)$ и $(y - x)$, а β_i — угол между векторами $(y^i - y)$ и $(x - y)$.

Найдем на отрезке xy такие две точки A и B , чтобы длина отрезков xA и yB была равна $\varepsilon |y - x|$. Проведем через точки A и B две гиперплоскости P_1 и P_2 , перпендикулярные к отрезку xy .

Выполнение условия (2) означает, что между гиперплоскостями P_1 и P_2 нет точек из множеств M и N и что гиперплоскость, проходящая через середину отрезка xy перпендикулярно к нему, является разделяющей для множеств M и N . Нетрудно видеть, что задача 1 является частным случаем задачи 3 при $\varepsilon = 1/2$, а задача 2 — при $\varepsilon = 0$.

Описываемый алгоритм дает возможность при $0 < \varepsilon \leq 1/2$ за конечное число шагов получать две точки $x \in \bar{M}$ и $y \in \bar{N}$, удовлетворяющие условию (1). Искомая разделяющая плоскость проводится через середину отрезка xy перпендикулярно к нему.

Перейдем к описанию алгоритма, предназначенного для нахождения точек x и y . Пусть x_i и y_i — i -е приближение к x и y . Обозначим члены бесконечного ряда $x^1, x^2, \dots, x^m, y^1, y^2, \dots, y^p, x^1, x^2, \dots, x^m, y^1, y^2, \dots, y^p, x^1, \dots$ через z_1, z_2, z_3, \dots .

В качестве первого приближения $x_1 y_1$ возьмем две произвольные точки из множеств M и N , т. е. $x_1 \in M$, $y_1 \in N$ (отсюда следует, что $x_1 \in \bar{M}$, $y_1 \in \bar{N}$).

Предположим, что точки $x_i \in \bar{M}$ и $y_i \in \bar{N}$ найдены. Определим следующее приближение x_{i+1}, y_{i+1} . Рассмотрим точку z_i (пусть для определенности $z_i \in M$). Тогда, полагая $y_{i+1} = y_i$, x_{i+1} , вычислим по формулам:

$$x_{i+1} = \begin{cases} x_i, & \text{если } (z_i - x_i, y_i - x_i) \leq \varepsilon (y_i - x_i, y_i - x_i), \\ z_i, & \text{если } (z_i - x_i, y_i - x_i) > \varepsilon (y_i - x_i, y_i - x_i) \\ & \text{и } (y_i - z_i, x_i - z_i) \leq 0, \\ z_i + [(z_i - y_i, z_i - x_i) / (z_i - x_i, z_i - x_i)] \times \\ & \times (x_i - z_i), \\ \text{если } (z_i - x_i, y_i - x_i) > \varepsilon (y_i - x_i, y_i - x_i) \\ & \text{и } (y_i - z_i, x_i - z_i) > 0. \end{cases}$$

Аналогично, если точка $z_i \in N$, то, полагая $x_{i+1} = x_i$, найдем y_{i+1} по формулам (3) (необходимо только x заменить на y , а y — на x с сохранением всех индексов).

Останов алгоритма возможен при двух условиях:

1. Расстояние между точками x_i и y_i все время уменьшается, вплоть до $(x_i - y_i, x_i - y_i) < \delta$, где $\delta > 0$ — малое число, зависящее от точности определения координат, и, возможно, от некоторых других условий задачи. При выполнении этого условия происходит останов и выдается ответ, что гиперплоскости, разделяющей множества M и N , не существует.

2. При некотором i ($i > m + p$) будут выполнены условия $x_{i-(m+p)} = x_i$ и $y_{i-(m+p)} = y_i$. Это соответствует тому, что для точек $x = x_i$ и $y = y_i$ выполнены условия (1). Происходит останов и по точкам x и y составляется уравнение искомой плоскости.

Пусть x_i и y_i — i -е приближение к x и y . Построим на отрезке x_i, y_i такие две точки A_i и B_i , чтобы длины отрезков $x_i A_i$ и $y_i B_i$ равнялись $\varepsilon |y_i - x_i|$. Проведем через эти точки две гиперплоскости P_i и Q_i , перпендикулярные к отрезку $x_i y_i$.

Рассмотрим точку z_i (пусть для определенности $z_i \in N$). Проверим, не противоречит ли появление точки z_i выполнению условий (1), т. е. выполнено ли неравенство $(z_i - y_i, x_i - y_i) \leq \varepsilon (x_i - y_i, x_i - y_i)$. Выполнение этого неравенства означает, что точка z_i лежит по ту же сторону от гиперплоскости Q_i , что и точка y_i или, в крайнем случае, на гиперплоскости Q_i . При этом не нужно менять точки x_i и y_i , т. е. полагаем $x_{i+1} = x_i$, $y_{i+1} = y_i$.

В противном случае, т. е. когда точки z_i и y_i лежат по разные стороны от гиперплоскости Q_i , точка z_i не удовлетворяет условию (1) и поэтому необходимо изменить y_i . При этом изменение должно произойти так, чтобы y_{i+1} принадлежало множеству \bar{N} и, кроме того, чтобы для точек z_i , $x_{i+1} = x_i$, y_{i+1} было выполнено соответствующее неравенство из условий (1). Однако ввиду ограниченности памяти ЭВМ запоминается только, что $x_i \in \bar{M}$, $y_i \in \bar{N}$ и $z_i \in \bar{N}$. Отсюда следует, что и весь отрезок $y_i z_i$ принадлежит множеству \bar{N} .

Таким образом точка y_{i+1} должна принадлежать отрезку $y_i z_i$ и должна быть возможно ближе к точке x_i . Такой точкой, очевидно, будет либо z_i [если только угол $y_i z_i x_i \geq 90^\circ$, что соответствует выполнению неравенства $(y_i - z_i, x_i - z_i) \leq 0$], либо основание перпендикуляра, опущенного из точки x_i на отрезок $y_i z_i$ [если угол $y_i z_i x_i$ острый, т. е. если выполнено неравенство $(y_i - z_i, x_i - z_i) > 0$]. В первом случае полагаем $y_{i+1} = z_i$, во втором y_{i+1} вычисляется по формуле

$$y_{i+1} = z_i + [(z_i - x_i, z_i - y_i) / (z_i - y_i, z_i - y_i)] (y_i - z_i).$$

Случай, когда $z_i \in M$, рассматривается аналогично.

Замечания

1. Можно доказать, что описанный алгоритм сходится при $0 < \varepsilon \leq 1/2$ за конечное число шагов. Доказательство этого факта аналогично доказательству, проведенному в работе [2].

2. Если для некоторого ε выяснилось, что не существует плоскости, разделяющей множества M и N , то, очевидно, ответ практически не изменится и при любом другом выборе ε .

этом, скажем, в каждую ячейку памяти машины М-220 можно записывать по пять координат исходных векторов. В частности, если векторы бинарны, т. е. координаты векторов принимают значения «0» или «1», в одну ячейку памяти машины М-220 может быть записано 45 координат. В этом случае в задачу блока 2 входит «разброс» координат «упакованного» вектора по ячейкам группы C , а также приписывание им соответствующего порядка (одинакового для всех координат). То же самое можно потребовать и от блока 1.

Блок 3 анализирует информацию о поступившем образе z_i . Если $z_i \in M$, то управление передается блоку 4, если же $z_i \in N$, то блок 3 засылает вектор a в ячейки B , а вектор b — в ячейки A и затем передает управление блоку 4. Одновременно блоку 8 сообщается, какому множеству принадлежал вектор z_i .

Блок 4 проверяет выполнение неравенства $(c - a, b - a) \leq (b - a, b - a)$. Если это неравенство выполнено, то управление передается блоку 8, в противном случае — блоку 5.

Блок 5 проверяет выполнение неравенства $(b - c, a - c) \leq 0$. Если это неравенство выполнено, то управление передается блоку 6, в противном случае — блоку 7.

Блок 6 записывает вектор c в ячейки A и передает управление блоку 8. Блок 7 вычисляет вектор a^* по формуле

$$a^* = c + [(c - b, c - a)/(c - a, c - a)](a - c),$$

записывает вектор a^* вместо вектора a в ячейки группы A и передает управление блоку 8. Блок 8 сразу передает управление блоку 9, если вектор z_i принадлежит множеству M . В противном случае блок 8 записывает вектор в ячейки B , а вектор b — в ячейки A^* .

Блок 9 проверяет условия останова. Если эти условия выполнены, то управление передается блоку 10; в противном случае — блоку 2, который подает следующий вектор z_{i+1} на обучение.

Блок 10 в зависимости от того, по какой причине произошел останов, либо выдает ответ, что искомой ги-

*) Заметим, что вектор a уже, вообще говоря, не тот, что был при работе, например, блока 3. Мы договорились через вектор a обозначать тот вектор, координаты которого в настоящее время записаны в ячейках группы A .

перп плоскости не существует, либо строит искомую гиперплоскость по точкам x и y .

Замечание. Распознавание вновь предъявленного вектора проводится по обычной для линейных обучаемых машин процедуре: выясняется, по какую сторону от построенной гиперплоскости находится вектор и в зависимости от этого он относится к тому или иному классу.

СПИСОК ЛИТЕРАТУРЫ

1. Якубович В. А. Некоторые общие теоретические принципы построения обучаемых «опознающих систем». В сб. «Вычислительная техника и вопросы программирования». Изд. ВЦ ЛГУ, 1965, вып. 4.
2. Козинец Б. Н. Об одном алгоритме обучения линейного персептрона. В сб. «Вычислительная техника и вопросы программирования». Изд. ВЦ ЛГУ, 1964, вып. 3.

И. Б. Мучник, Е. С. Петренко

Программы для решения задач распознавания образов методом потенциальных функций

Метод потенциальных функций возник в связи с решением задачи распознавания образов [1—4]. В ходе дальнейших исследований было установлено, что этот метод может успешно применяться и для решения ряда других задач (например, задачи интерполяции функции).

С помощью этого метода удалось выработать единый подход к решению задач статистической обработки многомерных массивов. Наиболее полно метод потенциальных функций описан в монографии [6], где дается теоретическое обоснование метода и описывается область его применения. Алгоритмы решения перечисленных задач методом потенциальных функций основаны на общей процедуре, что позволяет создать единую программу для их решения. Это означает, что с помощью небольших модификаций можно переходить от одного конкретного алгоритма к другому.

Ниже приводится описание программ, реализующих метод потенциальных функций (программы предназначены для обработки данных, заданных только в числовой форме). Метод потенциальных функций имеет две реализации. Соответственно приводятся две программы:

RECOGNITION PERCEPTRON — перцептронная реализация метода; и RECOGNITION POTENCIAL — потенциальная реализация метода.

1. ОБЩЕЕ ОПИСАНИЕ ПРОГРАММЫ ДЛЯ РЕШЕНИЯ ЗАДАЧ РАСПОЗНАВАНИЯ ОБРАЗОВ МЕТОДОМ ПОТЕНЦИАЛЬНЫХ ФУНКЦИЙ

Перечислим задачи, которые решаются этими программами:

- 1) задача 1: обучение машины распознаванию образов в детерминистской постановке;
- 2) задача 2: обучение машины распознаванию образов в вероятностной постановке;
- 3) задача 3: автоматическая классификация;
- 4) задача 4: интерполяция функций.

Для решения перечисленных задач методом потенциальных функций в «перцептронной» и «потенциальной» формах разработаны две программы, написанные на алгоритмическом языке ALGOL-60 и отлаженные с помощью транслятора TA-1M на ЭВМ М-220 (соответственно программы RECOGNITION PERCEPTRON, RECOGNITION POTENCIAL).

Предположим, что имеется последовательность векторов x_1, x_2, \dots, x_l размерности n ; назовем ее обучающей последовательностью [1—4]. Обе программы могут работать в двух режимах («обучение» и «экзамен»). В режиме «обучение» на основе обучающей последовательности строятся решающие функции. Процесс обучения прекращается при выполнении одного из условий:

а) обучающая последовательность просмотрена заданное число раз (S);

б) при очередном просмотре обучающей последовательности машина либо классифицирует все векторы правильно (задача 1), либо классифицирует их S_1 *) раз подряд точно так же, как и при предыдущем просмотре (задача 3).

При решении задач 2 и 4 для окончания обучения используется только условие а. После окончания обучения обучающая последовательность просматривается

*) В данной работе рассмотрено описание программ, реализующих алгоритмы решения задач распознавания образов. (Вопросы применения этих алгоритмов подробно рассмотрены в [5].)

еще один раз, формируя полученные результаты для вывода на печать (в программе RECOGNITION POTENTIAL реализовано только условие a).

В режиме «экзамен» на основе выработанных при обучении решающих функций выносится суждение о том, к каким классам принадлежат векторы, предъявляемые машине. Перестройка весовых коэффициентов и решающих векторов в этом режиме отсутствует.

Программы для решения первых трех задач оформлены в виде процедур с идентификаторами DET, PROBAB, SORT соответственно.

Формально задачу 4 удобно рассматривать как вариант задачи 2 распознавания K классов объектов при $K=1$; при этом в процедуре PROBAB осуществляются лишь небольшие изменения.

Помимо описанных в обеих программах имеются еще три процедуры (TRANSFORM, NORMA, ZACL), введение которых обусловлено следующими двумя обстоятельствами.

I. В соответствии с целями исследования часто целесообразно произвести предварительное преобразование входной последовательности. В программах предусмотрена возможность такого предварительного преобразования по одной из формул:

$$\tilde{x}_j^i = x_j^i / \sum_{j=1}^l x_j^i, \quad (1)$$

$$\tilde{x}_j^i = x_j^i / \sqrt{\sum_{j=1}^l (x_j^i)^2}, \quad (2)$$

$$\tilde{x}_j^i = (x_j^i - \bar{x}_j^i) / \sigma^i, \quad (3)$$

где x_j^i — значение i -го признака j -го вектора; \tilde{x}_j^i — значение того же i -го признака j -го вектора после нормирования; \bar{x}_j^i — значение i -го признака, усредненное по всем векторам обучающей последовательности; σ^i — оценка среднеквадратичного отклонения i -го признака, вычисленная по обучающей последовательности.

Процедура TRANSFORM реализует преобразование входной последовательности*) по формулам (1) и (2), процедура NORMA — по формуле (3).

*) Входная последовательность состоит из «обучающей» и «экзаменационной».

Преобразование по формуле (1) означает переход от абсолютных величин к относительным. При этом обычно имеется в виду, что природа всех признаков одинакова. В социологии примером задачи, в которой такое преобразование может представлять определенный интерес, является задача изучения структуры распределения свободного времени.

В этом случае $\sum_{j=1}^l x_j^i$ есть суммарное свободное время, отводимое совокупностью индивидов на i -е занятие, а \tilde{x}_j^i — доля свободного времени j -го индивида, которую составляет i -е занятие.

Преобразование по формуле (2) позволяет производить такое сравнение векторов, для которого увеличение значений всех признаков в определенное число раз не существенно. Пусть, например, x_j^i есть балл, которым j -й индивид оценивает i -ю характеристику условий его жизни на селе. В зависимости от степени его искренности оценки могут быть завышены или занижены в α раз. Если нас интересует не сама мера оценки условий жизни на селе, а только структура оценки по различным характеристикам, то с помощью формулы (2) можно избежать специального исследования степени искренности ответов.

Преобразование по формуле (3) позволяет переходить к безразмерным (отвлеченным) величинам, что существенно для сравнения векторов, заданных признаками разной природы. Кроме того, это преобразование уравнивает масштабы и диапазоны изменения векторов, а также приводит эти векторы к единому началу отсчета. Если при этом преобразовании окажется, что по некоторым признакам среднеквадратичное отклонение равно нулю, то рассматриваемая часть обучающей последовательности перестраивается: соответствующие признаки выбрасываются и размерность гиперпространства признаков понижается на число, равное числу таких признаков. Номера этих признаков выдаются на печать.

II. Часто во входной последовательности имеются «пропуски», т. е. для некоторых входных векторов отсутствуют измерения по одному или нескольким признакам. Для учета этого обстоятельства предусмотрена

процедура ZACL, которая на место «пропусков» ставит среднее значение соответствующего признака. Процедура NORMA на место «пропусков» заносит нули.

При работе программ входная последовательность предварительно преобразуется с помощью процедур NORMA или ZACL. После процедуры ZACL при желании может быть выполнена процедура TRANSFORM.

С помощью рекуррентных алгоритмов, описанных ниже, практически можно работать с входной последовательностью неограниченной длины. Для этого надо разбить эту последовательность на части (массивы), каждая из которых может быть размещена в оперативной памяти. Далее эти массивы по очереди вводятся в машину, после обработки очередного массива машина выдает все промежуточные данные, которые должны быть заданы как начальные условия при вводе в машину следующего массива *).

Программа допускает прерывание вычисления по условиям алгоритма. При этом все промежуточные данные, предусмотренные программой, выводятся на печать и в следующий раз надо ввести их вместе с очередной частью входной последовательности.

2. ОПИСАНИЕ ПРОГРАММЫ RECOGNITION PERCEPTRON

Решающие функции в этой программе находятся в форме гиперплоскости $(c_q, x) + a_q a_0$, где c_q — направляющий вектор; a_q — величина смещения гиперплоскости относительно начала координат в направлении вектора c_q ; a_0 — начальное значение этого смещения (если $a_0 = 0$, то все гиперплоскости проходят через начало координат).

Предполагается, что к началу работы программы:

а) в памяти машины имеются начальные векторы C_p^0 и константы a_p^0 , так что каждая такая пара соответствует определенному p -му классу**);

*) Такая обработка больших массивов данных легко реализуется автоматически, если весь исходный материал записан на приемлемом по времени считывания внешнем носителе. Существует модификация описываемой программы для работы с информацией, записанной на магнитной ленте.

**) Здесь и далее индекс класса будем обозначать строчной буквой, а название класса — соответствующей заглавной буквой.

б) входная последовательность x_1, x_2, \dots, x_l упорядочена таким образом, что сначала идут первые признаки всех векторов, затем вторые признаки, третьи и т. д. Кроме того, последовательность упорядочена по классам, т. е. сначала идут первые признаки векторов первого класса, затем первые признаки векторов второго класса и т. п. В программе указывается: n — размерность пространства X ; l — общее число векторов во входной последовательности; m_j — индекс класса, к которому относится j -й объект.

Как уже было сказано, если введенная в оперативную память машины последовательность векторов x_1, x_2, \dots, x_l представляет собой часть входной последовательности, на которой должны быть выработаны решающие функции, то программу следует преобразовать в процедуру (на языке ALGOL-60), которая используется последовательно столько раз, на сколько массивов оказалась разбита входная последовательность. В программе предусмотрено повторное прохождение вызванного в оперативную память массива обучающей последовательности заданное число раз. Следует помнить, что максимальный размер массива, который может быть вызван в оперативную память машины, обусловлен конкретными особенностями транслятора, размерностью гиперпространства, общим объемом оперативной памяти и т. п.

В процедурах ZACL и NORMA средние значения \bar{x}^i и среднеквадратические отклонения σ^i для каждого признака вычисляются рекуррентно. Пусть к моменту вычисления \bar{x}^i и σ^i программой обработано t векторов x_1, x_2, \dots, x_t и

вычислены значения $E_i = \sum_{p=1}^t x_p^i$ и $EE_i = \sum_{p=1}^t (x_p^i)^2$; тогда

при появлении l новых векторов $x_{t+1}, x_{t+2}, \dots, x_{t+l}$ значения \bar{x}_{t+l}^i и σ_{t+l}^i для i -го признака вычисляются по формулам

$$\bar{x}_{t+l}^i = \left(E_i + \sum_{p=t+1}^{t+l} x_p^i \right) / (t+l); \quad (4)$$

$$\sigma_{t+l}^i = \sqrt{\frac{EE_i + \sum_{p=t+1}^{t+l} \left[(x_p^i)^2 - \left(E_i + \sum_{p=t+1}^{t+l} x_p^i \right)^2 \right]}{t+l-1}}. \quad (5)$$

Здесь EE — обозначение переменной на языке ALGOL-60, а i — номер признака.

Рассмотрим, каким образом решается каждая из четырех перечисленных задач (решение задачи об интерполяции функции выполняется как вариант задачи 2 и описывается одновременно с ней).

Задача 1. При работе в режиме «обучения» распознаванию образов по информации о том, к какому классу принадлежит каждый вектор входной последовательности, находится набор решающих векторов c_1, c_2, \dots, c_k и констант a_1, a_2, \dots, a_k , таких, что

$$[(c_q, x) + a_q a_0] - [(c_p, x) + a_p a_0] > \varepsilon, \quad (6)$$

где $p=1, 2, \dots, K$; $q=1, 2, \dots, K$; $p \neq q$; $a_0 \geq 0$; $\varepsilon \geq 0$ для всех x , принадлежащих классу Q . Здесь величина ε определяет границу разделения; если $\varepsilon \neq 0$, то граница разделения классов задается интервалом $[+\varepsilon, -\varepsilon]$ и векторы, попавшие в этот интервал, будут считаться неопознанными, т. е. для них нельзя однозначно указать индекс класса.

Условие перестройки решающих векторов и констант при предъявлении очередного вектора x_r определяется неравенством

$$[(c_q^{r-1}, x_r) + a_q^{r-1} a_0] - [(c_p^{r-1}, x_r) + a_p^{r-1} a_0] \leq \varepsilon, \quad (7)$$

где x_r принадлежит классу Q , $\varepsilon \geq 0$ (условие должно выполняться для всех $p \neq q$).

Индекс r обозначает число просмотренных к данному моменту времени векторов всей входной последовательности (а не только той ее части, которая просматривается в этот момент).

Если условие перестройки не выполняется, то все решающие векторы и константы остаются без изменения, и рассматривается следующий объект входной последовательности.

Имеется два варианта перестройки решающих векторов и констант:

$$a) \ c_q^r = c_q^{r-1} + x_r, \ a_q^r = a_q^{r-1} + a_0, \quad (8)$$

где вектор x_r принадлежит Q -му классу; c_p^r и a_p^r (для $p \neq q$) не изменяются;

$$\begin{aligned} \text{б) } c_q^r &= c_q^{r-1} + x_r; \quad a_q^r = a_q^{r-1} + a_0; \quad c_g^r = c_g^{r-1} - x_r; \\ a_g^r &= a_g^{r-1} - a_0, \end{aligned} \quad (9)$$

где x_r принадлежит Q-му классу, а индекс g определяется из соотношения

$$[(c_g^{r-1}, x_r) + a_g^{r-1} a_0] = \max_{p \neq q} \{ (c_p^{r-1}, x_r) + a_p^{r-1} a_0 \}.$$

В программе второй вариант перестройки включает в себя первый как часть.

В режиме «экзамен» для определения принадлежности каждого нового вектора к одному из классов также используется неравенство (6), куда подставляются значения вектора c_p и константы a_p , вычисленные в процессе обучения.

Задача 2. В процессе обучения машине сообщается, к какому классу относится данный вектор (точно так же, как это происходит при обучении в задаче 1). Требуется построить такой набор решающих функций, с помощью которого любому из векторов можно было бы поставить в соответствие набор чисел, — вероятностей принадлежности этого вектора к каждому из возможных K классов. Эти функции определяются в линейной форме.

Перестройка решающих векторов и констант осуществляется по следующим формулам:

$$\begin{aligned} c_q^r &= c_q^{r-1} + \frac{A1}{B+r} (1 - \overline{sc}_q) x_r, \quad a_q^r = a_q^{r-1} + \frac{A1}{B+r} (1 - \overline{sc}_q) a_0, \\ c_{p \neq q}^r &= c_{p \neq q}^{r-1} + \frac{A1}{B+r} (0 - \overline{sc}_q) x_r, \quad a_{p \neq q}^r = a_{p \neq q}^{r-1} + \\ &+ \frac{A1}{B+r} (0 - sc_q) a_0, \end{aligned} \quad (10)$$

где $A1$ и B — числовые константы, определяющие скорость изменения решающих векторов и констант; вектор x_r относится к Q-му классу. В качестве переменной sc_i в этих формулах участвует выражение

$$sc_i = (c_i^{r-1}, x_r) + a_i^{r-1} a_0,$$

а операция „черта“ имеет следующий смысл:

$$\bar{A} = \begin{cases} D1, & \text{если } A < D1, \\ A, & \text{если } D1 \leq A \leq D2, \\ D2, & \text{если } A > D2, \end{cases}$$

причем $0 \leq D1 < D2 \leq 1$.

В отличие от предыдущей задачи перестройка решающих векторов и констант здесь производится для каждого вектора обучающей последовательности.

В режиме «экзамен» переменная sc_q определяет вероятность того, что вектор x_r относится к Q -му классу.

При решении задачи об интерполяции функции интервал $D1, D2$ определяет диапазон изменения значений функции, в котором производится ее интерполирование. В формулах, определяющих перестройку решающих векторов, выражение $(1 - sc_q)$ заменяется на $y_r - sc_1$, где y_r — значение интерполируемой функции для вектора x_r . Выписанный в оперативной памяти массив организован так, что вместо последнего признака всех векторов входной последовательности расположен ряд значений интерполируемой функции, соответствующих этой последовательности.

При решении задачи об интерполяции функции размерность векторов принимается равной $(n-1)$. Входная последовательность должна быть упорядочена следующим образом: сначала идут значения первого признака, затем — значения второго признака и т. д., далее — значения $(n-1)$ -го признака и значения функции y .

Задача 3. Алгоритм автоматической классификации векторов требует специального выбора начальных условий:

$$\begin{aligned} a_0 &= -1; \\ c_1 &= x_1; & a_1 &= |x_1|^2/2; \\ c_2 &= x_2; & a_2 &= |x_2|^2/2; \\ &\vdots & &\vdots \\ c_k &= x_k; & a_k &= |x_k|^2/2. \end{aligned}$$

где все векторы подмножества (x_1, x_2, \dots, x_k) входной последовательности строго различны.

При решении этой задачи решающие функции вырабатываются без сообщения машине информации о классах векторов обучающей последовательности.

Условие перестройки решающих векторов и констант определяется неравенством

$$[(c_g^{r-1}, x_r) + a_g^{r-1} a_0] - [(c_{p \neq g}^{r-1}, x_r) + a_{p \neq g}^{r-1} a_0] \geq \varepsilon. \quad (11)$$

Здесь ε имеет несколько иной смысл, чем в задаче 1. Хотя по-прежнему эта константа определяет точность разбиения, векторы, оказавшиеся в интервале $[+\varepsilon, -\varepsilon]$, относятся к особому $(k+1)$ -му классу.

Смысл неравенства (11) состоит в том, чтобы определить, имеются ли такие вектор c_g и константа a_g , при которых это неравенство оказывается справедливым для любых $p \neq g$. Если они найдутся, то вектор c_g и константа a_g перестраиваются, в противном случае перестройки не происходит.

Перестройка осуществляется следующим образом:

$$\begin{aligned} c_g^r &= c_g^{r-1} \left(1 - \frac{A1}{B+r} \right) + \frac{A1}{B+r} x_r, \\ a_g^r &= a_g^{r-1} \left(1 - \frac{A1}{B+r} \right) + \frac{A1}{B+r} [(c_g^{r-1}, x_r) + a_g^{r-1} a_0], \end{aligned} \quad (12)$$

где $A1 > 0, B > 0$.

Основные обозначения, используемые в программе

ММ — число векторов, которые были обработаны программой до ввода данной части входной последовательности. Если ММ=0, то программа начинает работать с нулевыми начальными векторами $C[P]$ ($P=1, 2, \dots, K$) и константами $A[P]$ ($P=1, 2, \dots, K$), если же ММ \neq 0, то программа вводит массивы начальных векторов C и констант A , а также массивы EX и EE (предполагается, что массивы C, A, EX, EE ранее вычислены с помощью этой же программы);

МК — число векторов, которые обрабатываются при данном включении программы;

N — размерность пространства X ;

K — число классов;

S — необходимое число просмотров обрабатываемой части входной последовательности;

RS — число шагов, которое было сделано программой ранее при обработке ММ векторов до данного включения программы; если ММ=0, то надо задать RS=1;

NN — константа, указывающая, какое предварительное преобразование необходимо произвести с входной последовательностью;

RR — константа, задающая режим работы; если $RR=0$, то программа работает в режиме «обучение», если $RR=1$, — в режиме «экзамен»;

PP — константа, разрешающая ($PP=1$) или запрещающая ($PP=0$) печать результатов вычисления;

V — константа, указывающая вид «поощрения» в задаче 1;

A1, B1 — константы в формулах (10)–(12);

TEST — константа, задающая номер задачи, которая будет решаться при данном включении программы: если $TEST=1$, то решается задача 1, если $TEST=2$, то — задача 2, если $TEST=3$, то решается задача автоматической классификации. Как уже было сказано выше, для решения задачи интерполяции функции надо решать задачу обучения машины распознаванию образов в вероятностной постановке с числом классов, равным единице ($K=1$);

TYPE — константа, задающая правило перестройки решающих векторов C и констант A при данном включении программы; когда $TYPE=1$, используется перестройка векторов C и констант A по формулам (8) или (9), определенным для решения задачи 1; при $TYPE=2$ используется перестройка векторов C и констант A по формулам (10), определенным для решения задачи 2; если же $TYPE=3$, то используется перестройка векторов C и констант A по формулам (12), определенным для решения задачи автоматической классификации векторов;

EPS — число, меняющее (как параметр) условие перестройки векторов C и констант A в задаче 1 и в задаче 4 [см. неравенства (7) и (11)];

D1 — нижняя допустимая граница в формулах (10);

D2 — верхняя допустимая граница в формулах (10);

A0 — значение начального смещения; если $MM \neq 0$ и $TEST=3$, то $A0=-1$;

$EX[1:N]$ — массив сумм $\sum_P x_p^i$, накопленных для всех признаков при обработке массивов векторов до данного включения программы;

$EE[1:N]$ — аналогично полученный массив сумм $\sum_P (x_p^i)^2$;

$C[1:K, 1:N]$ — массив решающих векторов;

$A[1:K]$ — массив решающих констант;

$M[1:MK]$ — массив, указывающий номера классов для всех векторов обрабатываемой части входной последовательности;

$X[1:N, 1:MK]$ — массив обрабатываемой части входной последовательности;

$SC[1:K]$ — массив решающих функций для каждого объекта обрабатываемой части входной последовательности;

$B[1:MK]$ — «библиотека» ответов: если J-й объект входной последовательности верно опознан при последнем просмотре обрабатываемой части входной последовательности, то $B[J]=1$; в противном случае $B[J]=-1$; если же J-й объект не опознан, $B[J]=0$;

$D[1:MK]$ — массив значений функции вероятности принадлежности J-го вектора к $M[J]$ классу для всех векторов обрабатываемой части входной последовательности;

MN — число неопознанных объектов;

CH — число ошибок;

SH — число отказов.

В программе оформлены следующие процедуры и переключатели:

NORMA, ZACL, TRANSFORM — процедуры, определяющие необходимое преобразование массива обрабатываемой части входной последовательности;

INCOND — процедура задания начальных условий в задаче 3;

SCALAR — процедура, вычисляющая массив SC[1 : K] решающих функций [см. формулу (10)]. В задачах 1 и 3 следует задавать $D1 = 10^{18}$ и $D2 = -10^{18}$, точнее самое большое и самое малое из возможных в машине чисел;

STIMULATION — процедура перестройки векторов C и констант A;

DET, PROBAB, SORT — процедуры, определяющие условие перестройки в задачах 1, 2, 3 соответственно. Все эти процедуры являются процедурами без параметров. Массивы X, EX, EE, M, C, A, SC и числа A1, B, MM, MK, N, K являются нелокальными параметрами этих процедур.

Процедуры DET, PROBAB, SORT вычисляют номера векторов C, которые должны изменяться процедурой STIMULATION и подготавливают результаты для выдачи на печать.

RESULT — переключатель, указывающий место в программе, куда передается управление при работе в режиме «экзамен».

3. ОПИСАНИЕ ПРОГРАММЫ RECOGNITION POTENCIAL

Эта программа для решения задач 1, 2, 3 и 4 строит решающие функции в классе нелинейных функций. В этом случае алгоритмы для решения задач распознавания образов методом потенциальных функций не являются рекуррентными.

В качестве элементов для построения решающих функций используется так называемая потенциальная функция («потенциал»). Обычно для этой цели применяется некоторая функция расстояния между двумя векторами.

В настоящей программе этой функцией является

$$\varphi(x, y) = 1/[1 + \alpha R^2(x, y)],$$

где $R(x, y)$ — евклидово расстояние между векторами x и y ; $\alpha > 0$.

Теперь для формирования решающей функции и использования ее при «экзамене» необходимо хранить в памяти машины всю обучающую последовательность векторов x_1, x_2, \dots, x_l . Поэтому такая реализация метода потенциальных функций носит название машинной реализации.

Любой Q -й класс в обучающей последовательности характеризуется суммарной потенциальной функцией

$$K(q, y) = \sum_{x_j \in Q} a_{qj} \varphi(x_j y), \quad (13)$$

где суммирование ведется по всем объектам, принадлежащим классу Q . Набор чисел a_{qj} строится в процессе обучения; число a_{qj} представляет собой весовой коэффициент j -го объекта, принадлежащего Q -му классу обучающей последовательности.

Считается, что всякий вектор принадлежит Q -му классу, если

$$K(q, x_j) = \max_p K(p, x_j), \quad (14)$$

где $p = 1, 2, \dots, k$.

Вид решающей функции в задаче 3 отличается от выражения (14) (он будет рассмотрен ниже). Для простоты построения программы оказалось удобным рассматривать задачу 3 несколько обособленно. В программе оформлено два блока. В первом блоке при помощи процедур POTENCIAL, DET, PROBAB решаются задачи 1, 2 и 4. Во втором блоке при помощи процедур с идентификатором POTENCIAL (отличной от процедуры первого блока с тем же идентификатором) и идентификатором SORT решается задача 3.

Как и первая программа, программа RECOGNITION POTENCIAL может работать в двух режимах. В режиме «обучение» в машину вводится обучающая последовательность, состоящая из объектов x_1, x_2, \dots, x_l . В режиме «экзамен» в машину вводится последовательность, состоящая из векторов $x_1, x_2, \dots, x_l, x_{l+1}, \dots, x_{l+g}$, где векторы x_1, \dots, x_l принадлежат обучающей последовательности, а векторы $x_{l+1}, x_{l+2}, \dots, x_{l+g}$ — экзаменационной.

В программе оформлена процедура-функция с идентификатором FUNC(Q, J), вычисляющая «потенциал» между q -м и j -м объектами в соответствии с формулой (13):

$$f(x_q, x_j) = 1/(1 + \alpha R_{qj}^2), \quad (15)$$

где $\alpha > 0$; $R_{qj}^2 = \sum_{i=1}^n (x_j^i - x_q^i)^2$ (n — размерность вектора x).

Задача 1. Условие, определяющее, что на r -м шаге необходимо произвести перестройку весовых коэффициентов, задается неравенством

$$\left(\sum_{x_q \in P} a_{pq}^{r-1} f(x_q, x_r) - \sum_{x_q \in G} a_{gq}^{r-1} f(x_q, x_r) \right) \leq \varepsilon, \quad (16)$$

где $\varepsilon \geq 0$; $G \neq P$; x_r — j -й объект из обучающей последовательности, предъявленный машине на r -м шаге. Вектор x_r принадлежит классу P ; a_{pq}^{r-1} — значение весового коэффициента q -го вектора из P -го класса, вычисленное к $(r-1)$ -му шагу.

Если условие перестройки (16) не выполняется, то весовые коэффициенты остаются без изменения и машине предъявляется очередной вектор из обучающей последовательности. Имеется два варианта перестройки весовых коэффициентов на r -м шаге:

$$a) \quad a_{pj}^r = a_{pj}^{r-1} + \Delta; \quad \Delta > 0,$$

$$a_{gj}^r = a_{gj}^{r-1} \quad \text{для } g \neq p, \quad (17)$$

$$g = 1, 2, \dots, p-1, p+1, \dots, K;$$

$$б) \quad a_{pj}^r = a_{pj}^{r-1} + \Delta, \quad a_{gj}^r = a_{gj}^{r-1} - \Delta, \quad (18)$$

где a_{gj}^{r-1} определяется из соотношения:

$$\sum_{x_q \in G} a_{gq} f(x_q, x_j) = \max_s \left\{ \sum_{x_q \in S} a_{sq}^{r-1} f(x_q, x_j) \right\}$$

$$(s = 1, 2, \dots, p-1, p+1, \dots, K).$$

В программе второй вариант перестройки весовых коэффициентов включает первый как часть.

Обучение заканчивается или когда при очередном просмотре обучающей последовательности все векторы опознаны правильно или в конце заранее заданного S -го просмотра.

Задача 2. Перестройка весовых коэффициентов осуществляется отдельно для каждого предъявляемого машине вектора. Если на r -м шаге машине предъявляется

j -й вектор, принадлежащий P -му классу, то весовые коэффициенты перестраиваются по формулам:

$$a_{pj}^r = a_{pj}^{r-1} + \frac{1}{r} \left[1 - \sum_{x_q \in P} \overline{a_{pq}^{r-1} f(x_q, x_j)} \right]; \quad (19)$$

$$a_{gj}^r = a_{gj}^{r-1} - \frac{1}{r} \sum_{x_q \in G} \overline{a_{gq}^{r-1} f(x_q, x_j)}. \quad (20)$$

Здесь $q=1, 2, \dots, p-1, p+1, \dots, K$; $G \neq P$; операция «черта» имеет тот же смысл, что и в формуле (10). Суммирование в формуле (19) ведется по всем векторам обучающей последовательности, принадлежащим классу P ; в формуле (20) суммирование производится по всем векторам класса G . Обучение заканчивается после просмотра обучающей последовательности заданное число S раз.

В режиме «экзамен» выражение

$$K_p = \sum_{x_q \in P} a_{pq} f(x_p, x_q) \quad (21)$$

означает вероятность того, что j -й объект относится к P -му классу.

Для решения задачи 4 используется процедура с идентификатором PROBAB, в которой выполняются изменения, аналогичные описанным для программы RECOGNITION PERCEPTRON.

Задача 3. Для решения задачи 3 в программе RECOGNITION POTENCIAL оформлен второй блок, в отличие от программы RECOGNITION PERCEPTRON, где для решения этой задачи оформлена процедура с идентификатором SORT.

Процедура с идентификатором POTENCIAL, локализованная во втором блоке описываемой программы, используется для вычисления значений «потенциала» для j -го вектора, относительно векторов P -го класса по формуле

$$K_p(x_j) = \sum_{\substack{x_q \in P \\ q \neq j}} f(x_q, x_j), \quad (22)$$

где суммирование ведется по всем векторам класса P из обучающей последовательности, за исключением j -го.

Напомним, что в задаче автоматической классификации в режиме «обучение» начальные условия выбираются так, чтобы начальное разбиение множества векторов, подлежащих классификации, содержало хотя бы по одному вектору каждого класса из обучающей последовательности, причем эти векторы равны между собой и не равны нулю.

Процесс обучения автоматической классификации состоит из двух этапов. Первый этап заключается в том, что каждый вектор обучающей последовательности индексируется, т. е. относится к одному из возможных классов (1, 2, ..., K). При этом для j -го еще неопознанного вектора вычисляются значения «потенциала» относительно векторов каждого класса по формуле (22). Затем для каждого класса вычисляется функция «потенциал класса» по формуле

$$S_p^{r-1} = \sum_{x_g \in P} K_p(x_g) = \sum_{x_q \in P} \sum_{x_g \in P} f(x_q, x_g), \quad (23)$$

где суммирование производится по всем векторам, отнесенным на $(r-1)$ -м шаге к классу P. Далее j -й вектор относится к P-му классу по условию

$$\Delta_p = \max_q \{\Delta q\} = \max_q \left\{ \frac{[(c_q^{r-1} - 1)K_q(x_j) - 2c_q^{r-1}S_q^{r-1}]}{[(c_q^{r-1} + 1)c_q^{r-1}(c_q^{r-1} - 1)]} \right\}, \quad (24)$$

где $q=1, 2, \dots, K$; c_q^{r-1} — число векторов, отнесенных к классу Q на $(r-1)$ -м шаге. Если на r -м шаге вектор x_j относится к P-му классу, то осуществляется перестройка решающих функций S_p и констант c_p по следующим формулам:

$$S_p^r = S_p^{r-1} + K_p(x_j), \quad c_p^r = c_p^{r-1} + 1. \quad (25)$$

На втором этапе предполагается, что в машине кроме обучающей последовательности введены константы c_p и решающие функции S_p ($p=1, 2, \dots, K$), полученные на первом этапе. На этом этапе обучающая последовательность просматривается несколько раз; при этом каждый раз корректируются значения S_p и c_p . Если все векторы обучающей последовательности при очередном просмотре распознаются точно так же, как при предыдущем, обучение заканчивается. Обучение заканчивается и в том случае, если этого не случилось,

но обучающая последовательность была просмотрена заранее заданное число раз S .

Условие перестройки решающих функций S_p и констант c_p на r -м шаге в режиме «переиндексация», т. е. при повторных просмотрах обучающей последовательности, определяется выражением

$$\begin{aligned} \Delta_p &= \max_q \{\Delta_q\} = \\ &= \max_q \{[c_q^{r-1} - 1]K_q(x_j) - 2c_q^{r-1}S_q^{r-1}/[(c_q^{r-1} + 1)c_q^{r-1}(c_q^{r+1} - 1)] - \\ &\quad - [c_i^{r-1}K_i(x_j) - 2(c_i^{r-1} - 1)S_i^{r-1}]/[c_i^{r-1}(c_i^{r-1} - 1)(c_i^{r-1} - 2)]\}, \end{aligned} \quad (26)$$

где i — индекс класса, к которому вектор x_j был отнесен на $(r-1)$ -м шаге; остальные обозначения те же, что и в формуле (24).

Если условие (26) выполняется, то осуществляется перестройка по следующим формулам:

$$\begin{aligned} S_p^r &= S_p^{r-1} + K_p(x_j), \quad c_p^r = c_p^{r-1} + 1; \\ S_i^r &= S_i^{r-1} - K_i(x_j), \quad c_i^r = c_i^{r-1} - 1. \end{aligned} \quad (27)$$

При работе в режиме «экзамен» в памяти машины должны находиться:

- а) последовательность векторов $x_1, x_2, \dots, x_l, x_{l+1}, \dots, \dots, x_{l+g}$;
- б) потенциальные функции, построенные в режиме «обучение» $S_p, p=1, 2, \dots, K$;
- в) последовательность констант $c_p, p=1, 2, \dots, K$;
- г) массив индексов классов каждого из векторов обучающей последовательности x_1, x_2, \dots, x_l , полученный в режиме «обучение».

Каждый новый вектор $x_j, j=l+1, \dots, l+g$ относится к P -му классу, определяемому по формуле (26).

Основные обозначения, используемые в программе

- K — число классов;
- MO — число векторов в обучающей последовательности;
- ME — длина всей входной последовательности, включая обучающую и экзаменационную последовательность;
- N — размерность гиперпространства;
- S — максимальное число просмотров обучающей последовательности;

RR — режим работы программы; если $RR=0$, то выполняется режим «обучение» в задачах 1, 2, 4 и режим первого этапа обучения в задаче 3; при $RR=1$ — режим второго этапа обучения в задаче 3, а при $RR=2$ — режим «экзамен»;

PP — константа, разрешающая ($PP=1$) или запрещающая ($PP=-C$) печать результатов вычисления;

TEST — номер решаемой задачи; если $TEST=1$, то решается задача 1; если $TEST=2$, то — задача 2; если $TEST=2$ и $K=1$, то выполняется задача 4; если $TEST=3$, то — задача 3;

EPS — константа, определяющая условие перестройки весовых коэффициентов в задаче 1;

D1 — нижняя допустимая граница в формулах (10);

D2 — верхняя допустимая граница в формулах (11);

DELTA — константа, определяющая перестройку решающих функций в задаче 1;

ALPHA — константа для вычисления решающей функции по формуле (15);

X[1 : N, 1 : MK] — массив входной последовательности векторов (в режиме «обучение» $MK=MO$, а в режиме «экзамен» — $MK=ME$);

A[1 : K, 1 : MO] — массив весовых коэффициентов в задачах 1, 2, 4 (этот массив должен вводиться в режиме «экзамен»);

V — константа, определяющая формулу перестройки весовых коэффициентов в задачах 1, 2, 4;

M[1 : MK] — массив индексов классов каждого вектора из входной последовательности;

SPOT[1 : K] — массив потенциальных функций [см. формулу (23)]; этот массив вводится на втором этапе обучения и при режиме «экзамен» в задаче 3;

CH[1 : K] — массив чисел векторов в каждом классе; вводится в задаче 3 на втором этапе обучения и при «экзамене»;

R — число векторов, просмотренных в режиме «обучение» (число шагов);

EX[1 : N] — массив средних значений всех признаков векторов обучающей последовательности [см. формулу (3)];

EE[1 : N] — массив сумм квадратов значений всех признаков векторов обучающей последовательности [см. формулу (3)];

A[1 : K, 1 : MK] — массив весовых коэффициентов;

CH — число неверно распознанных векторов (число ошибок) при последнем просмотре обучающей последовательности;

SH — число неопознанных векторов при последнем просмотре обучающей последовательности (число отказов);

B[1 : MK] — «библиотека» ответов при последнем просмотре обучающей последовательности; если J-й вектор опознан правильно, то $B[J]=1$, в противном случае $B(J)=0$; если же $B[J]=-1$, то J-й объект опознан неверно;

POT[1 : K] — массив значений функции «потенциал» [см. формулу (22)] для каждого J-го вектора обучающей последовательности.

4. О ВЫБОРЕ ФОРМЫ ПОТЕНЦИАЛЬНОЙ ФУНКЦИИ

Программа RECOGNITION POTENCIAL позволяет за счет небольшого изменения использовать различные виды потенциальных функций, описанные ниже. Для этого имеются следующие возможности:

а) можно использовать различные формулы для подсчета расстояния между векторами, в частности, если компоненты векторов равны «0» или «1», то естественно вычислять расстояние по Хеммингу (оно равно числу несовпадающих компонент сравниваемых векторов). В задачах, где вектор характеризуется набором логических переменных, такой выбор расстоянием естествен. Однако расстояние по Хеммингу может быть использовано и при работе с векторами, которые характеризуются числовыми компонентами. Это бывает целесообразно, например, в таких случаях, когда многие из компонент имеют сложные масштабные шкалы, которые заранее известны;

б) одной из простейших форм потенциальной функции, с помощью которой строится разделяющая гиперповерхность в форме гиперплоскости, является скалярное произведение векторов:

$$(x^{(1)}x^{(2)}) = \sum_{i=1}^n x_i^{(1)} x_i^{(2)}. \quad (28)$$

Реализация этой формулы получается за счет модификации блока, осуществляющего вычисление расстояния между векторами.

При желании строить разделяющую гиперповерхность, уравнение которой, кроме линейной части, содержало бы и более высокие степени произведений различных компонент, можно использовать формулу

$$\sum_{s=1}^p a_s (x^{(1)}x^{(2)}), \quad (29)$$

где константы a_s , p — параметры, подбираемые в ходе эксперимента.

Потенциальные функции в форме, которая зависит только от расстояния между векторами в гиперпространстве признаков, отличается большой наглядностью и поэтому при изучении конкретных задач целесообразно искать такое преобразование признаков, при котором эта форма наиболее проста. Важной особенностью такой формы потенциальной функции является ее симметричный характер, т. е. все признаки «одинаково существенны» для построения разделяющей гиперповерхности. В практике, однако, часто возникают случаи, когда

априори ясно, что признаки не равноценны, но заранее не известны их весовые коэффициенты *). Если весовые коэффициенты h_i определены, то расстояния между векторами можно вычислить по формуле

$$R^2 = \sum_{i=1}^n h_i [\mathbf{x}_i^{(1)} - \mathbf{x}_i^{(2)}]^2, \quad (30)$$

пользуясь при этом всеми описанными выше алгоритмами решения задач.

Возможно и наоборот, попытаться «уравновесить» роль различных параметров, используя, например, следующую процедуру: построить на обучающей последовательности матрицу парных коэффициентов «близости» между параметрами. Обозначим эту матрицу через λ_{ij} .

С помощью такой матрицы можно получить новую систему параметров:

$$u_i = \sum \lambda_{ij} x_j. \quad (31)$$

Очевидно, что параметры, найденные по формуле (31), становятся более «уравновешенными».

5. ИНСТРУКЦИЯ ПО ВВОДУ И ВЫВОДУ ИНФОРМАЦИИ ПРИ РАБОТЕ ПРОГРАММ RECOGNITION

Вставить в читающее устройство колоду перфокарт, состоящую из двух частей, следующих одна за другой. Первая часть — перфокарты с текстом программы на языке ALGOL-60. Вторая часть — массив исходных данных для работы программы, порядок расположения которых описан ниже для каждой программы отдельно.

1. Инструкция к программе Recognition perceptron

Порядок расположения исходных данных

1. При решении задачи 1 в режиме «обучение» с нулевыми начальными условиями в устройство ввода должны быть вставлены перфокарты со следующей информацией.

*) Существует ряд работ, в которых обсуждаются возможности получать оценки весовых коэффициентов на основе данных обучающей последовательности [6].

| Номер перфокарты | Содержание | Примечание |
|---------------------|------------|--|
| 1 | $MM = 0$ | |
| 2 | МК | Число векторов, которые будут обрабатываться при данном включении программы |
| 3 | N | Размерность гиперпространства (число признаков) |
| 4 | K | Число классов |
| 5 | S | Число повторных просмотров обучающей последовательности |
| 5-A | $S1 = 0$ | |
| 6 | $RS = 1$ | |
| 7 | NN | Константа, указывающая вид преобразования, которому будет подвергнута исходная последовательность векторов |

Если $NN = 1$, то $\tilde{x}_j^i = (x_j^i - \bar{x}^i) / \sigma^i$; если $NN = 2$, то $\tilde{x}_j^i = x_j^i / \left(\sum_{j=1}^{МК} x_j^i \right)$; если $NN=3$, то $\tilde{x}_j^i = x_j^i \left[\sum_{j=1}^{МК} (x_j^i)^2 \right]^{-1/2}$, где x_j^i , \tilde{x}_j^i —

соответственно старое и новое значения i -го признака j -го объекта входной последовательности; \bar{x}^i — среднее значение i -го признака; σ^i — среднеквадратическое отклонение по i -му признаку.

Если $NN=0$, то выполняется процедура с идентификатором ACL (см. описание программы)

| | | |
|----|-----------------|---|
| 8 | $RR = 0$ | |
| 9 | $PP = 1$ | |
| 10 | $VV = 0$ | |
| 11 | V | Константа, указывающая вид „поощрения“ *) |
| 12 | $TEST = 1$ | |
| 13 | $TYPE = 1$ | |
| 14 | $EPS = 0$ | |
| 15 | $D1 = -10^{18}$ | |
| 16 | $D2 = 10^{18}$ | |
| 17 | $AO = 1$ | |
| 18 | $A1 = 0$ | Константы в формулах (10) и (12); |
| 19 | $B1 = 0$ | |

*) Если $V=0$, то перестройка решающих векторов производится по формуле (8), а при $V=1$ — по формуле (9).

На каждой из этих двадцати перфокарт должно быть набито два числа в десятичной форме с плавающей запятой, первое из которых — значение константы, а второе — значение этой константы с признаком контрольной суммы ($k\Sigma$).

Начиная с двадцатой перфокарты должны идти массивы перфокарт в следующем порядке:

$M[1:MK]$ — массив, указывающий индексы классов для каждого вектора обрабатываемой части входной последовательности;

$X[1:N, 1:MK]$ — массив векторов (исходных данных) обрабатываемой части входной последовательности.

Последней перфокартой каждого массива должна идти перфокарта с контрольной суммой этого массива.

2. При решении задачи 1 в режиме «обучение» с ненулевыми начальными условиями порядок перфокарт тот же, что и в п. 1, за исключением 1-й перфокарты, где MM — число векторов, которые были обработаны ранее, и 7-й перфокарты, где RS — суммарное число шагов, которое было сделано программой ранее за все циклы при обработке MM векторов.

Начиная с 20-й перфокарты должны идти следующие массивы:

$EX[1:N]$ — массив сумм $\sum_{p=1}^{MK} x_p^i, i = 1, 2, \dots, N$, вычисленный

для всех признаков ранее, т. е. при просмотре MM векторов;

$EE[1:N]$ — массив сумм $\sum_{p=1}^{MK} (x_p^i)^2, i = 1, 2, \dots, N$, вычисленными

для всех признаков ранее;

$C[1:K, 1:N]$ — массив решающих векторов, вычисленный ранее,

$A[1:K]$ — массив решающих констант, вычисленный ранее.

Последней перфокартой каждого массива должна идти перфокарта с контрольной суммой этого массива.

Далее следуют те же массивы $M[1:MK]$ и $X[1:N, 1:MK]$, что и в п. 1.

3. При решении задачи 1 в режиме «экзамен» исходные данные должны быть рассмотрены в том же порядке, что и в п. 2, за исключением 8-й перфокарты ($RR=1$).

4. При решении задачи 2 в режиме «обучение» с нулевыми начальными условиями в исходных данных по сравнению с п. 1 должны быть сделаны следующие изменения:

| Номер перфокарты | Содержание | Примечание |
|------------------|------------|----------------------------------|
| 12 | TEST = 2 | |
| 13 | TYPE = 2 | |
| 15 | D1 = 0 | |
| 16 | D2 = 1 | |
| 18 | A1 | Значения констант в формуле (10) |
| 19 | B1 | |

Если на 4-й перфокарте задать $K=1$, то будет решаться задача об интерполяции функции. В этом случае значения интерполируемой функции должны помещаться на место последнего признака во входной последовательности векторов.

5. При решении задачи 2 в режиме «обучение» с ненулевыми начальными условиями должны быть сделаны по сравнению с п. 4 изменения, которые описаны в п. 2.

6. При решении задачи 2 в режиме «экзамен» исходные данные должны быть подобраны в том же порядке, что и в п. 5 за исключением 8-й перфокарты (RR=1).

7. При решении задачи 3 в режиме «обучение» с нулевыми начальными условиями исходные данные должны быть подготовлены, как описано в п. 1, со следующими изменениями:

| Номер перфокарты | Содержание | Примечание |
|------------------|------------|---|
| 5-A | S1 | Константа, определяющая условие прекращения цикла обучения *) |
| 12 | TEST = 3 | |
| 13 | TYPE = 3 | |
| 17 | AO = -1 | |
| 18 | A1 } | |
| 19 | B1 } | Значения констант в формуле (12) |

Начиная с 18-й перфокарты должен располагаться массив X[1 : N, 1 : MK].

8. При решении задачи 3 в режиме «обучение» с ненулевыми начальными условиями исходные данные должны быть подобраны в порядке, описанном в п. 2, за исключением 12-й перфокарты — TEST=3, 13-й — TYPE=3 и 17-й — AO=-1.

9. При решении задачи 3 в режиме «экзамен» исходные данные должны быть подобраны, как это указано в п. 8 за исключением 8-й перфокарты (RR=1).

Печать результатов вычислений

В режиме «обучение» при решении всех описанных задач (1—4) программа печатает следующую информацию (в перечисленном порядке):

1. Массив сумм $\sum_{p=1}^{MK} x_p^i, i = 1, 2, \dots, N$ (EX [1:N]).
2. Массив сумм $\sum_{p=1}^{MK} (x_p^i)^2, i = 1, 2, \dots, N$ (EE [1:N]).

2-A. Индексы признаков, для которых при NN=1 $\sigma_i=0$.

3. Массив всех компонент среднего вектора \bar{x} для просматриваемого массива векторов.

4. Массив значений дисперсии для каждого признака просматриваемого массива векторов σ_i .

5. Массив решающих векторов (C[1 : K, 1 : N]).

6. Массив решающих констант (A[1 : K]).

7. Число векторов, обработанных программой к данному моменту времени (MM).

8. Суммарное число шагов, которое было сделано алгоритмом к данному моменту (RS).

*) Если S1=0, т. е. цикл прекращается по условию а, если S1=1 — по условию б (см. § 1),

Кроме того, в режиме «обучение» при решении задачи 1 дополнительно печатается:

9. Число отказов на просматриваемой последовательности векторов.

10. Число ошибок на просматриваемой последовательности векторов.

11. Библиотека ответов, содержащая МК кодов: если на j -м месте от начала массива (библиотеки) стоит «1», то j -й вектор опознан верно, если же на j -м месте стоит «-1», то j -й вектор опознан неверно; если на j -м месте — «0», произошел отказ, т. е. j -й объект неопознан («0» может быть при $EPS \neq 0$).

При решении задачи 2 в режиме «обучение» дополнительно печатаются:

12. Значения интерполируемой векторной функции вероятности принадлежности каждого вектора к каждому из возможных K классов.

При решении задачи 3 — «обучение» дополнительно печатается:

13. Число неопознанных векторов.

14. Для каждого вектора из входной последовательности печатается:

а) индекс класса, к которому данный вектор отнесен; если напечатан «0», то это значит, что данный вектор не опознан («0» может быть при $EPS \neq 0$);

б) индекс вектора во входной последовательности j ;

в) массив значений решающих функций для данного вектора $SC[1:K]$.

В режиме «экзамен» печатается информация, указанная в пп. 9—14 (различная в зависимости от вида решаемой задачи).

II. Инструкция к программе Recognition Potencial

Порядок расположения исходных данных

1. При решении задачи 1 в режиме «обучение» должны быть подготовлены перфокарты со следующей информацией.

| Номер перфокарты | Содержание | Примечание |
|------------------|------------------------|--|
| 1 | K | Число классов |
| 2 | MO | Число векторов в обучающей последовательности |
| 3 | ME | Число векторов во всей входной последовательности (обучающая плюс экзаменационная последовательность); |
| 4 | N | Размерность пространства (число признаков) |
| 5 | S | Число просмотров обучающей последовательности |
| 6 | RR = 0 | |
| 7 | PP = 1 | |
| 8 | TEST = 1 | |
| 9 | EPS = 0 | |
| 10 | D1 = -10 ¹⁸ | |

| | | |
|----|----------------|---|
| 11 | $D2 = 10^{18}$ | |
| 12 | DELTA | Константа Δ в формулах (17) и (18) |
| 13 | ALPHA | Константа α в формуле (15) |
| 14 | NN | Константа, указывающая вид преобразования входной последовательности объектов (см. инструкцию к программе RECOGNITION PERCEPTOR). |

На каждой из четырнадцати перфокарт пробиваются два числа в десятичной форме с плавающей запятой, первое из которых — значение константы, а второе — значение этой константы с признаком контрольной суммы ($h\Sigma$).

Начиная с 15-й перфокарты расположен массив $X[1:N, 1:MK]$. Затем идет перфокарта, на которой записана контрольная сумма этого массива. Следующая перфокарта: V — константа, определяющая вид перестройки весовых коэффициентов; при $V=0$ весовые коэффициенты перестраиваются по формуле (17), при $V=1$ — по формуле (18).

Далее расположен массив $M[1:MO]$, а затем перфокарта с контрольной суммой этого массива.

2. При решении задачи 1 в режиме «экзамен» перфокарты должны быть подготовлены так, как описано в п. 1, за исключением 6-й перфокарты ($RR=1$).

Следом за массивом $X[1:N, 1:ME]$ должны располагаться массивы:

$EX[1:N]$ — массив сумм $\sum_{p=1}^{MO} x_p^i, i = 1, 2, \dots, N$, вычисленный на

обучающей последовательности объектов;

$A[1:K, 1:MK]$ — массив весовых коэффициентов, вычисленных в режиме «обучение».

Последней перфокартой каждого массива должна идти перфокарта с контрольной суммой этого массива. Затем идут перфокарты с константой V и массивом $M[1:MO]$ (как указано в п. 1).

3. При решении задачи 2 в режиме «обучение» го сравнению с п. 1 надо сделать следующие изменения: в 8-й перфокарте $TEST=2$; в 10-й — $D1=0$; в 11-й — $D2=1$.

4. При решении задачи 2 в режиме «экзамен» перфокарты подбираются, как и в п. 2, за исключением 8-й перфокарты — $TEST=2$, 10-й — $D1=0$, 11-й — $D2=1$.

5. При решении задачи об интерполяции функции в режиме «обучение» перфокарты должны быть подобраны так, как это указано в п. 3, за исключением 1-й перфокарты ($K=1$).

6. При решении задачи об интерполяции функции в режиме «экзамен» перфокарты подбираются в соответствии с п. 4, за исключением 1-й перфокарты ($K=1$).

7. При решении задачи 3 на первом этапе обучения порядок расположения перфокарт соответствует описанному в п. 1, за исключением 8-й перфокарты — $TEST=3$ и 12-й — $DELTA=0$.

Информация, расположенная после массива $X[1:N, 1:MO]$ в п. 1, должна отсутствовать, т. е. последней перфокартой будет контрольная сумма массива.

8. При решении задачи 3 на втором этапе обучения перфокарты подготавливаются в соответствии с п. 7, за исключением того, что начиная с 15-й перфокарты должны следовать массивы:

$M[1:MO]$ — массив индексов классов векторов обучающей последовательности, вычисленный на первом этапе обучения;

$SPOT[1:K]$ — массив потенциальных функций (23), вычисленный на первом этапе обучения;

$CH[1:K]$ — массив, указывающий число векторов, попавших в каждый p -й класс ($p=1, 2, \dots, K$) на первом этапе обучения.

Затем следует массив $X[1:N, 1:MO]$. Последняя перфокарта каждого массива должна содержать контрольную сумму этого массива. Кроме того, в 6-й перфокарте $RR=1$.

9. При решении задачи 3 в режиме «экзамен» перфокарты подготавливаются в соответствии с п. 8, за исключением 6-й перфокарты ($RR=2$).

В этом режиме массив $X[1:N, 1:ME]$ представляет собой всю входную последовательность векторов.

Печать результатов вычислений

При решении задач 1, 2, 4 в режиме «обучение» программа печатает следующие результаты:

- 1) число шагов, сделанное программой к данному моменту (R);
- 2) число просмотров обучающей последовательности S ;
- 3) массив среднеквадратичных отклонений для всех признаков σ_i , $i=1, 2, \dots, N$;

- 4) массив всех компонент среднего вектора x^i , $i=1, 2, \dots, N$;

4а) массив сумм $\sum_{j=1}^{MO} (x_j^i)^2$, $i=1, 2, \dots, N$;

- 5) массив весовых коэффициентов $A[1:K; 1:MK]$.

При решении задачи 1 в режиме «экзамен» печатается:

- 6) число ошибок CH ;
- 7) число отказов SH ;
- 8) библиотека ответов $B[1:MK]$;
- 9) значения потенциальных функций для каждого вектора.

При решении задач 2 и 4 в режиме «экзамен» печатается:

- 10) значение интерполируемой функции для каждого вектора.

При решении задачи 3 на первом и на втором этапах обучения печатаются:

- 11) индексы класса каждого вектора обучающей последовательности $M[1:MO]$;

- 12) потенциальные функции (23);
- 13) количество векторов в каждом классе $CH[1:K]$;
- 14) массив среднеквадратических отклонений σ_i ;
- 15) массив средних значений \bar{x}^i ;

15а) массив сумм $\sum_{j=1}^{MO} x_j^i$.

При решении задачи 3 в режиме «экзамен», кроме того, печатается:

- 16) значение «потенциалов» всех классов (15) для каждого вектора обучающей последовательности.

RECOGNITION PERCEPTRON;

BEGIN

INTEGER I,J,G,P,MM,MN,MK,N,K,S, RS, R ,SH,CH,NN,RR,PP, V,

VV,TEST, TYPE ;

REAL EPS,D1,D2,AO,MAX,MAX2;

POO42 (MM, MK, N,K, S, RS,NN,RR,PP,VV,V,TEST,TYPE,EPS ,

D1,D2,AO) ;

BEGIN ARRAY EX,EE [1:N] ,C[1:K,1:N],A,SC[1:K],X[1:N,1:MK]
, D[1 :MK];

INTEGER ARRAY B, M[1:MK];

PROCEDURE NORMA;

BEGIN REAL DIS; FOR I:=1 STEP 1 UNTIL N DO

BEGIN R:=MM;

FOR J:=1 STEP 1 UNTIL MK DO

IF X[I, J] \neq_{10+18} THEN

BEGIN R:=R +1; EX[I]:=EX[I]+X[I,J];

EE[I] :=EE[I] + X[I,J] \uparrow 2

END;

DIS:=SQRT ((EE [I]-EX[I] \uparrow 2/R)/(R-1));

IF DIS=0 THEN B[I] :=I ELSE BEGIN

FOR J:=1 STEP 1 UNTIL MK DO

IF X[I,J] \neq_{10+18} THEN X[I,J] :=0

ELSE X[I,J] :=(X[I,J]-EX[I]/R)/DIS

END END; MM: =MM+MK

END;

PROCEDURE ZACL;

BEGIN FOR I:=1 STEP 1 UNTIL N DO

BEGIN R:=MM; FOR J:=1 STEP 1 UNTIL MK DO

IF X[I, J] \neq_{10+18} THEN

BEGIN R:=R+1; EX[I] :=EX[I]+X[I,J];

EE[I] :=EE[I]+X[I,J] \uparrow 2;

END;

FOR J:=1 STEP 1 UNTIL MK DO

IF X[I,J] \neq_{10+18} THEN X[I,J] :=EX[I]/R

END; MM:=MM+MK

END;

PROCEDURE TRANSFORM;

BEGIN FOR I:=1 STEP 1 UNTIL N DO

```

FOR J:=1 STEP 1 UNTIL MK DO
  IF NN=2 THEN X[I,J] := X[I,J] / EX[I]
  ELSE X[I,J] := X[I,J] / SQRT(EE[I])
END;
PROCEDURE INCOND;
BEGIN INTEGER ARRAY IR[1:K];
  FOR P:=1 STEP 1 UNTIL K DO
    BEGIN IR[P] := 1; A[P] := 0;
      FOR I:=1 STEP 1 UNTIL N DO C[P,I] := 0
    END; G:=0; SH:=CH; IF CH=1 THEN SH:=CH-1;
L1: SH:=SH+1; IF SH>MK THEN GOTO L2;
    MN:=0;
    FOR I:=1 STEP 1 UNTIL N DO
      IF X[I,SH]=0 THEN MN:=MN+1; IF MN=N THEN
        GOTO L1; FOR P:=1 STEP 1 UNTIL G DO
        BEGIN MN:=0; FOR I:=1 STEP 1 UNTIL N DO
          IF C[P,I]=X[I,SH] THEN MN:=MN+1;
          IF MN=N THEN GOTO L1
        END; G:=G+1; FOR I:=1 STEP 1 UNTIL N DO
        BEGIN C[G,I] := X[I,SH]; IR[G] := SH;
          A[G] := A[G] + C[G,I]^2
        END; A[G] := A[G] / 2;
        IF G<K THEN GOTO L1;
      P1041 (IR)
    END;
L2:
PROCEDURE SCALAR;
BEGIN FOR P:=1 STEP 1 UNTIL K DO
  BEGIN SC[P] := 0; FOR I:=1 STEP 1 UNTIL N DO
    SC[P] := SC[P] + C[P,I] * X[I,J] + A[P] * AO;
    IF SC[P]<D1 THEN SC[P] := D1 ELSE
    IF SC[P]>D2 THEN SC[P] := D2
  END
END;
PROCEDURE STIMULATION;
BEGIN REAL ALPHA;
  IF TYPE=1 THEN
    BEGIN FOR I:=1 STEP 1 UNTIL N DO
      C[M[J],I] := C[M[J],I] + X[I,J];
      A[M[J]] := A[M[J]] + AO; IF V=I THEN

```

```

        BEGIN A[G] := A[G]-AO;
        FOR I:=1 STEP 1 UNTIL N DO
            C[G,I]:=C[G,I]-X[I,J]
        END
    END; IF TYPE=1 THEN GOTO LEND;
    IF VV=1 THEN ALPHA:=1/LN(R) ELSE ALPHA:=1/R;
    IF TYPE=3 THEN GOTO L3; IF K≠1 THEN GOTO LK;
    A[M [ J]]:=A[M [ J]]+ALPHA*(X[SH,J]-SC[M [ J]])*AO
); FOR I:=1 STEP 1 UNTIL N DO
    C[M [ J],I]:=C[M [ J],I]+ALPHA*( X[SH,J]-SC[M [ J]]
) *X[I,J];
    GOTO LEND;
LK:    A[M [ J]]:=A[M [ J]]+ALPHA*(1-SC[M [ J]])*AO;
    FOR I:=1 STEP 1 UNTIL N DO
        C[M [ J], I]:=C[M [ J],I]+ALPHA*(1-SC[M [ J]])*X[I,
J]; FOR P:=1 STEP 1 UNTIL K DO IF P≠M[J] THEN
        BEGIN A[P]:=A[P]-ALPHA*SC [P]*AO;
            FOR I:=1 STEP 1 UNTIL N DO
                C[P,I]:=C[P,I]-ALPHA*SC [P]*X[I,J]
            END; GOTO LEND;
L3:    A[G]:=A[G]*(1-ALPHA)+ALPHA*SC [G];
    FOR I:=1 STEP 1 UNTIL N DO
        C[G,I]:=C[G,I]*(1-ALPHA)+ALPHA*X[I,J];
LEND:  END;
    PROCEDURE DET;
    BEGIN
LL:    CH:=SH:=0; FOR J:=1 STEP 1 UNTIL MK DO
        BEGIN R:=R+1; SCALAR; IF M[J]≠1 THEN
            BEGIN MAX:=SC[I]; G:=1
            END ELSE
            BEGIN MAX:=SC[2]; G:=2
            END; FOR P:=1 STEP 1 UNTIL K DO
                IF P≠M[J]^MAX < SC[P] THEN
                    BEGIN MAX:=SC[P]; G:=P
                    END; IF (SC [ M[J]]-SC[G]) < EPS THEN
                        BEGIN CH:=CH+1; B[J]:=-1;
                            IF (SC[M [ J]]-SC[G] ) > 0 THEN
                                BEGIN SH:=SH+1; B[J]:=0
                                END; IF RR=0 THEN STIMULATION
                                END ELSE B [J]:=1
                            END; IF RR=OACH*GA/(R-RS)/MK < S THEN GOTO LL; CH:=CH-EN

```

```

END;
PROCEDURE PROBAB;
BEGIN
LL:  IF K=1 THEN
      BEGIN SH:=N; N:=N-1; FOR J:=1 STEP 1 UNTIL MK DO
            BEGIN R:=R+1; SCALAR;
                  IF RR=0 THEN STIMULATION
                  ELSE D [J]:=SC[M [J]]
            END; N:=SH
      END ELSE FOR J:=1 STEP 1 UNTIL MK DO
            BEGIN R:=R+1; SCALAR; IF RR=0 THEN STIMULATION ELSE
                  D[J]:=SC[M [J]]
            END; IF RR=OΛ(R-RS)/MK<S THEN GOTO LL;
END;

PROCEDURE SORT;
BEGIN IF RR=0 THEN FOR J:=1 STEP 1 UNTIL MK DO M[J]:=0;
LL:  MN:=0; FOR J:=1 STEP 1 UNTIL MK DO
      BEGIN R:=R+1; SCALAR; IF SC[1]>SC[2] THEN
            BEGIN MAX:=SC [1]; G:=1; MAX2:=SC[2]
            END ELSE
            BEGIN MAX:=SC[2]; G:=2; MAX2:=SC[1]
            END; FOR P:=3 STEP 1 UNTIL K DO IF MAX<SC[P] THEN
            BEGIN MAX2:=MAX; MAX:=SC[P]; G:=P
            END; IF (MAX-MAX2) > EPS THEN
            BEGIN IF RR=0 THEN STIMULATION;
                  IF M[J]= G THEN CH:=0 ELSE CH:=1; M[J] :=G
            END ELSE MN:=MN+1
            END; IF RR=OΛ(R-RS)/MK<S ^CH=1 THEN GOTO LL
END;

IF MM=0 THEN POO42(EX,EE,C,A) ELSE FOR I:=1 STEP 1 UNTIL N DO
BEGIN CH:=0; EX[I] :=EE[I]:=0;
      FOR P:= 1 STEP 1 UNTIL K DO
            BEGIN A[P]:=0; FOR I:=1 STEP 1 UNTIL N DO C[P,I] :=0
            END;
END; IF TEST43 THEN POO42(M); POO42(X);
IF NN>1 THEN TRANSFORM ELSE IF NN=1 THEN NORMA ELSE ZACL;
P1041(EX,EE,B); FOR I:=1 STEP 1 UNTIL N DO
BEGIN EE[I]:=SQRT((EE[I]-EX[I]2/MM)/(MM-1));
      EX[I]:=EX [I]/MM
END; P1041(EX,EE);

```



```

P:=0; FOR I:=1 STEP 1 UNTIL N DO IF EE[I]=0 THEN
BEGIN FOR G:=1 STEP 1 UNTIL N-P DO
    FOR J:=1 STEP 1 UNTIL MK DO X[G-P,J]:=
        X[G-P+1,J]; P:=P+1
END; N:=N- P; IF RR=1 THEN GOTO REZULT; R:=RS;
IF TEST=1 THEN DET ELSE IF TEST=2 THEN PROBAB ELSE
BEGIN
    IF MM=MK THEN INCOND; AO:=-1; SORT
END;
RR:=1; IF PP=1 THEN P1041(C,A,MM,R);
REZULT: IF TEST=1 THEN BEGIN DET; IF PP=1 THEN
BEGIN P1041(SH,CH,B); FOR J:=1 STEP 1 UNTIL MK DO
    BEGIN SCALAR; P1041(SC)
    END
END END ELSE IF TEST=2 THEN
BEGIN PROBAB; P1041( D)
END ELSE
BEGIN SORT; IF PP=1 THEN
    BEGIN P1041(MN); FOR J:=1 STEP 1 UNTIL MK DO
        BEGIN SCALAR; P1041(M[J], SC)
        END
    END
END
END
END
END;

```

RECOGNITION POTENCIAL;

BEGIN

INTEGER I,J,P,Q,G,K,MK,N,R,S,NN,RR,PP,TEST,MO,C,ME;

REAL EPS,D1,D2,DELTA,ALPHA;

POO42(K,MO,ME,N,S,RR,PP,TEST,EPS,D1,D2,DELTA,ALPHA,NN);

IF RR=0 THEN

BEGIN C:=1; MK:=MO

END ELSE

BEGIN MK:=-ME; C:=MO+1

END;

BEGIN ARRAY EX,EE[1:N], POT[1:K], X[1:N,1:MK];

INTEGER ARRAY M[1: MK];

REAL PROCEDURE FUNC(Q,J); VALUE Q,J; INTEGER Q,J;

BEGIN REAL RAB; RAB:=0; FOR I:=1 STEP 1 UNTIL N DO

RAB:=RAB+(X[I,J]-X[I, Q])²;

RAB:=1/(1+ALPHA*xRAB); FUNC:=RAB

END;

POO42(X); IF RR=0 THEN FOR I:=1 STEP 1 UNTIL N DO

EX[I]:=EE[I]:=0 ELSE POO42(EX,EE);

FOR I:=1 STEP 1 UNTIL N DO

BEGIN R:=0; FOR J:=1 STEP 1 UNTIL MK DO IF X[I,J]≠₁₀₊₁₈ THEN

BEGIN R:=R+1; EX[I]:=EX[I]+X[I,J]

END; EX[I]:=EX[I]/R;

FOR J:=1 STEP 1 UNTIL MK DO

IF X[I,J]≠₁₀₊₁₈ THEN EE[I]:=EE[I]+(EX[I]-X[I,J])²;

EE[I]:=SQRT(EE[I]/(R-1)); P1041(EE[I]); EE[I]:=0;

FOR J:=1 STEP 1 UNTIL MK DO

BEGIN IF X[I,J]=₁₀₊₁₈ THEN X[I,J]:= EX[I];

IF NN=1 THEN X[I,J]:=(X[I,J]-EX[I])/EE[I];

IF NN=2 THEN X[I,J]:=X[I,J]/(EX[I]*R);

END; IF NN=3 THEN

BEGIN FOR J:=1 STEP 1 UNTIL MK DO EE[I]:=EE[I]+X[I,J]⁴

2;X[I,J]:=X[I, J]/SQRT (EE[I])

END;

END; IF TEST≠3 THEN

BEGIN REAL MAX; INTEGER CH,SH,V; ARRAY A[1:K, 1:MK], B[C:MK];

; PROCEDURE POTENCIAL;

BEGIN FOR P:=1 STEP 1 UNTIL K DO

BEGIN POT[P]:=0; FOR Q:=1 STEP 1 UNTIL MK DO

```

    IF M[Q] = PAQ/J THEN
    POT[P] := POT[P] + A[P,Q] * FUNC(Q,J)
    END
END;
PROCEDURE DET;
BEGIN
LL:  CH:=SH:=0; FOR J:=C STEP 1 UNTIL MK DO
    BEGIN R:=R+1; POTENCIAL; IF M[J]≠1 THEN
        BEGIN MAX:=POT[1]; G:=1
        END ELSE
        BEGIN MAX:=POT[2]; G:=2
        END; FOR P:=1 STEP 1 UNTIL K DO
            IF P≠M[J] ∧ MAX < POT[P] THEN
                BEGIN MAX:=POT[P]; G:=P
                END; IF (POT[M[J]] - POT[G]) < EPS THEN
                    BEGIN CH:=CH+1; B[J]:=-1; IF RR=0 THEN
                        BEGIN A[M[J], J] := A[M[J], J] + DELTA;
                        IF V=1 THEN A[G,J] := A[G,J] - DELTA
                        END ELSE
                        IF (POT[M[J]] - POT[G]) > 0 THEN
                            BEGIN SH:=SH+1; B[J]:=0
                            END;
                        END ELSE B[J]:=1
                    END; IF RR=0 AND CH≠OAR/MK < S THEN GOTO LL;
                    IF RR=1 THEN
                        BEGIN P1041(CH,SH,B); FOR J:=C STEP 1 UNTIL MK DO
                            BEGIN POTENCIAL; P1041(POT)
                            END
                        END
                    END;
END;
PROCEDURE PROBAB;
BEGIN
LL:  IF K=1 THEN
    BEGIN SH:=N; N:=N-1; FOR J:=C STEP 1 UNTIL MK DO
        BEGIN R:=R+1; POTENCIAL;
        FOR P:=1 STEP 1 UNTIL K DO
            IF POT[P] > D1 THEN POT[P]:=D1 ELSE
            IF POT[P] < D2 THEN POT[P]:=D2;
            IF RR=0 THEN
                A[M[J], J] := A[M[J], J] + (X[SH,J] - POT[M[J]])/R

```

```

        ELSE B[J]:=POT[M[J]]
    END; N:=SH
    FOR J:=C STEP 1 UNTIL MK DO
        BEGIN R:=R+1; POTENCIAL;
            FOR P:=1 STEP 1 UNTIL K DO IF POT[P]<D1 THEN
                POT[P]:=D1 ELSE IF POT[P]>D2 THEN POT[P]:=D2;
                IF RR=0 THEN
                    BEGIN A[M[J],J]:=A[M[J],J]+(1-POT[M[J]])/R;
                        FOR Q:=1 STEP 1 UNTIL K DO
                            IF Q=M[J] THEN A[Q,J]:=A[Q,J]-POT[Q]/R
                        END ELSE
                            B[J]:= POT[M[J]]
                    END; IF RR=0/R/MK<S THEN GOTO LL;
                    IF RR=1 THEN P1041(B)
                END;
            IF RR=1 THEN P0042(A) ELSE
                FOR P:=1 STEP 1 UNTIL K DO FOR J:=1 STEP 1 UNTIL MK
                    A[P,J]:=0; P0042(V,M);
                    IF TRST=1 THEN DET ELSE PROBAB;
                    IF RR=0 THEN
                        BEGIN P1041(R,R/MK,EX,A);
                            END
                        END ELSE
                            BEGIN REAL DELTAP,DELTAL,MAX; INTEGER PI;
                                ARRAY SPOT[1:K]; INTEGER ARRAY CH[1:K];
                                PROCEDURE POTENCIAL;
                                    BEGIN FOR P:=1 STEP 1 UNTIL K DO
                                        BEGIN POT[P]:=0; FOR Q:=1 STEP 1 UNTIL MO DO
                                            IF M[Q]=PAQ4J THEN POT[P]:=POT[P]+FUNC(Q,J)
                                        END
                                    END; IF RR=0 THEN
                                        BEGIN FOR P:=1 STEP 1 UNTIL K DO
                                            BEGIN SPOT[P]:=0; CH[P]:=2; M[P]:=P
                                            END; C:=K+1
                                        END ELSE P0042(M,SPOT,CH); R:=0;
                                        FOR J:=C STEP 1 UNTIL MK DO
                                            BEGIN R:=R+1; POTENCIAL; IF RR=1 THEN
                                                BEGIN I:=M[J]; PI:=0;
                                                    DELTAL:=CH[I]*POT[I]-2*(CH[I]-1)*SPOT[I];
                                                    IF CH[I]>2 THEN DELTAL:=DELTAL/6 ELSE

```

12

12.3

```

      DELTA I:=DELTA I/CH[I]x(CH[I]-1)x(CH [I]-2)
END;
FOR Q:=1 STEP 1 UNTIL K DO
  BEGIN P:=CH[Q]; DELTA P:=(P-1)xPOT[Q]-2xPxSPOT[Q];
    IF P<2 THEN DELTA P:=DELTA P/6 ELSE
      DELTA P:=DELTA P/(P+1)xPx(P-1);
    IF RR=1 THEN DELTA P:=DELTA P-DELTA I; IF Q=1 THEN
      BEGIN MAX:=DELTA P; G:=1
    END ELSE IF MAX<DELTA P THEN
      BEGIN MAX:=DELTA P; G:=Q
    END
  END; IF RR<2 THEN
    BEGIN SPOT[G]:=SPOT[G]+POT[G]; CH[G]:=CH[G]+1;
      IF RR=1 THEN
        BEGIN SPOT[I]:=SPOT[I]-POT[I]; CH[I]:=CH[I]-1
          ; IF I<G THEN PI:=1
        END
      END; M[J]:=G
    END; IF RR=1/PI=1/AR/MK<S THEN GOTO LL; P1041(M);
    IF RR<2 THEN P1041(SPOT,CH,EX,EE) ELSE
      FOR J:=1 STEP 1 UNTIL MO DO
        BEGIN POTENCIAL; P1041(POT)
      END
    END
  END
END
END
END;

```

СПИСОК ЛИТЕРАТУРЫ

1. Айзерман М. А., Браверман Э. М., Розоноэр Л. И. Теоретические основы метода потенциальных функций в задаче об обучении автоматов разделению входных ситуаций на классы. «Автоматика и телемеханика», 1964, т. XXV, № 6
2. Айзерман М. А., Браверман Э. М., Розоноэр Л. И. Вероятностная задача об обучении автоматов распознаванию классов и метод потенциальных функций. «Автоматика и телемеханика», 1964, т. XXV, № 9.
3. Айзерман М. А., Браверман Э. М., Розоноэр Л. И. Метод потенциальных функций в задаче о восстановлении характеристики функционального преобразователя по случайно наблюдаемым точкам. «Автоматика и телемеханика», 1964, т. XXV, № 12.
4. Браверман Э. М. Метод потенциальных функций в задаче обучения машины распознаванию образов без учителя. «Автоматика и телемеханика», 1966, т. XXV, № 10.

5. Неймарк Ю. И., Баталова З. С. Опыт использования быстродействующей вычислительной машины для медицинской диагностики, прогнозирования исхода оперативного вмешательства и выбора оптимального метода лечения. Ученые записки НИИ ПМК и ф-та ВМиК Горьковского Государственного университета. Горький, 1967.

6. Айзерман М. А., Браверман Э. М., Розоноэр Л. И. Метод потенциальных функций в теории обучения машин. Изд-во «Наука», 1970.

В. Н. Фомин

Об одном рекуррентном алгоритме нахождения гиперплоскости, разделяющей выпуклые множества на единичном кубе

Часто задача обучения распознаванию образов может быть интерпретирована как задача создания алгоритма построения гиперплоскости, разделяющей непересекающиеся множества векторов. Существует большое число рекуррентных алгоритмов (т. е. алгоритмов с минимальным ограничением на память системы, см. [3]), позволяющих решить указанную задачу [1—5].

Обычно разделяющая гиперплоскость определяется не в самом множестве векторов, а в так называемом спрямляющем пространстве, в котором отображаются исходные множества с помощью « α -элементов» [3]. Часто спрямляющее пространство совпадает с множеством вершин некоторого единичного куба, а исходные множества отображаются в некоторые подмножества его вершин. В этом случае полезно иногда построить (если это возможно) разделяющую гиперплоскость, направляющий вектор которой состоит из нулей и единиц.

Ниже описан алгоритм, позволяющий построить указанную гиперплоскость при условии, что «оболочки» рассматриваемых множеств не пересекаются. (Точное определение оболочки множества дается ниже и является обобщением понятия выпуклой оболочки множества в евклидовом пространстве).

Идея алгоритма проста: определяется пара точек, расстояния между которыми совпадают с расстояниями между оболочками бинарных векторов; вектор, определяемый этими точками, является искомым. Аналогичный алгоритм для случая выпуклых множеств в евклидовом пространстве описан в работе [5]. Сразу отметим, что

требование непересечения оболочек множеств бинарных векторов является, в определенном смысле, более сильным ограничением, чем требование непересечения выпуклых оболочек множеств в случае евклидова пространства. Это обстоятельство можно рассматривать как плату за простоту алгоритма.

Условимся о некоторых обозначениях и понятиях. Буквами x, y, z, w (с нижними индексами или без них) обозначим векторы, компоненты которых есть 0 и 1. Через $x \oplus y$ обозначим вектор z , полученный покомпонентным сложением векторов x и y по модулю 2, т. е. если $x = (x^{(1)}, \dots, x^{(N)})$, $y = (y^{(1)}, \dots, y^{(N)})$, то $z = (z^{(1)}, \dots, z^{(N)})$, где $z^{(i)} = x^{(i)} \oplus y^{(i)}$, $i = 1, 2, \dots, N$.

Аналогично, через $x \wedge y$ обозначим вектор z , полученный покомпонентным умножением векторов x и y , т. е.

$$z^{(i)} = x^{(i)} y^{(i)} \quad (i = 1, 2, \dots, N).$$

Нормой $|x|$ вектора x называется сумма содержащихся в нем единиц $|x| = \sum_{i=1}^N x^{(i)}$. Расстояние $\rho(x, y)$

(расстояние по Хеммингу) и скалярное произведение (x, y) векторов x и y определяются следующим образом:

$$\rho(x, y) = |x \oplus y|; \quad (x, y) = |x \wedge y|.$$

Через e_i , $i = 1, 2, \dots, N$ обозначим базисные векторы (i -я координата вектора e_i равна единице, остальные координаты есть нули). Очевидно, всякий вектор x однозначно может быть представлен в виде

$$x = \sum_{i=1}^N x^{(i)} e_i,$$

где $x^{(i)}$ — компоненты вектора x .

Пусть известны два вектора x и y . Введем понятие о векторе, расположенном между векторами x и y . Для этого образуем вектор $z = x \oplus y$, который может быть представлен в виде

$$z = \sum_{i=1}^N z^{(i)} e_i,$$

Условимся, что вектор w расположен между векторами x и y , если его можно представить в виде

$$w = x \oplus \sum' z^{(i)} e_i.$$

Штрих у суммы означает, что суммирование происходит по некоторому подмножеству индексов (в частности, это подмножество может быть пустым, и тогда $w = x$, или совпадать со всем множеством векторов $i = 1, 2, \dots, N$, и тогда $w = y$). Геометрическая интерпретация этого понятия очевидна. Действительно, если рассматривать векторы x и y как вершины единичного куба, то множество всех векторов, заключенных между x и y , состоит из всех вершин (включая x и y), через которые проходит точка при движении от x к y вдоль ребер куба всевозможными кратчайшими путями*). Множество Q на единичном кубе называется *подкубом* (кубом меньшей размерности), если из условия $x_1 \in Q$, $x_2 \in Q$ следует, что множеству Q принадлежат все вершины, заключенные между вершинами x_1 и x_2 .

Оболочкой Q множества F на единичном кубе называется подкуб наименьшей размерности, содержащий это множество.

В дальнейшем будут рассматриваться множества F_1 и F_2 , относительно которых предполагается, что их оболочки Q_1 и Q_2 не пересекаются**).

Расстоянием $\rho(F_1, F_2)$ между множествами на единичном кубе называется величина

$$\rho(F_1, F_2) = \min |x \oplus y|, \quad (1)$$

где \min берется по всем $x \in F_1$ и $y \in F_2$. Основное содержание понятия оболочки множества на единичном кубе заключается в следующем: обозначим через x_0, y_0 пару векторов ($x_0 \in Q_1, y_0 \in Q_2$), для которых

$$\rho(x_0, y_0) = \rho(Q_1, Q_2) > 0,$$

т. е. x_0, y_0 — точки, расстояние между которыми и определяет расстояние между оболочками множеств. При этом имеют место неравенства:

$$\begin{aligned} |x_0 \oplus x| &< |y_0 \oplus x|, & x \in Q_1; \\ |x_0 \oplus y| &> |y_0 \oplus y|, & y \in Q_2; \end{aligned} \quad (2)$$

*) Кратчайшим называется путь, при движении вдоль которого точка проходит минимальное число ребер куба.

**) В частности, может случиться, что $F_1 = Q_1, F_2 = Q_2$.

Отсюда нетрудно получить следующее свойство, эквивалентное (2):

$$\begin{aligned}(x, z_0) &= (x_0, z_0) && \text{при } x \in Q_1, \\ (y, z_0) &= (y_0, z_0) && \text{при } y \in Q_2,\end{aligned}\tag{3}$$

где

$$z_0 = x_0 \oplus y_0.\tag{4}$$

Таким образом, для разделения множеств F_1 и F_2 достаточно построить по формуле (4) вектор z_0 , который и определяет искомую гиперплоскость согласно формулам (3) *).

Сформулируем рекуррентный алгоритм для нахождения векторов x_0, y_0 , реализующих расстояние между подкубами Q_1 и Q_2 . Векторы x_1 и y_1 выбираются произвольно, $x_1 \in F_1, y_1 \in F_2$. Допустим, что на n -м шаге получены векторы $x_n, y_n, x_n \in Q_1, y_n \in Q_2$, которые являются n -й итерацией искомых векторов x_0, y_0 .

Пусть системе предъявлен вектор z . Образует вектор

$$w_n = z \wedge (x_n \oplus y_n) \oplus x_n \wedge y_n.\tag{5}$$

Тогда

$$\begin{cases} x_{n+1} = w_n, \\ y_{n+1} = y_n, \end{cases} \quad \text{если } z \in F_1,$$

$$\begin{cases} x_{n+1} = x_n, \\ y_{n+1} = w_n, \end{cases} \quad \text{если } z \in F_2.$$

Пусть z_1, z_2, \dots — бесконечная последовательность векторов, составленная из векторов множеств F_1 и F_2 . Если в этой последовательности каждый вектор множеств F_1 и F_2 повторяется бесконечное число раз, то нетрудно доказать, что начиная с некоторого номера N выполняется равенство

$$\begin{cases} x_{n+1} = x_n, \\ y_{n+1} = y_n, \end{cases} \quad n \geq N,$$

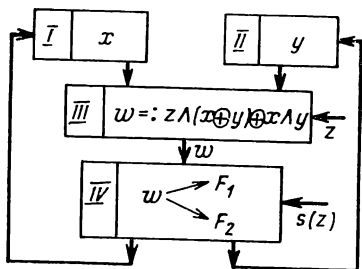
причем расстояние между оболочками Q_1 и Q_2 множеств F_1 и F_2 достигается на векторах x_N и y_N .

*) Отметим, что вектор z_0 определяется по оболочкам Q_1, Q_2 единственным образом, хотя векторы x_0, y_0 могут определяться неоднозначно.

Формулу (5) можно представить в следующих эквивалентных формах:

$$w_n = x_n \oplus (x_n \oplus z) \wedge (x_n \oplus y_n) = y_n \oplus (y_n \oplus z) \wedge \\ \wedge (x_n \oplus y_n) = z \wedge x_n \oplus x_n \wedge y_n \oplus z \wedge y_n.$$

Блок-схема алгоритма изображена на рисунке. В блоках I, II хранятся соответственно итерация векторов x и y , реализующих расстояние между оболочками множеств F_1 и F_2 . В качестве начальных итераций векторов x_1, y_1 используются произвольные векторы из множеств F_1, F_2 . В блоке III происходит вычисление вектора w_n по формуле (5). Результат записывается в блок IV, где устанавливается принадлежность вектора к оболочке множества F_1 или F_2 . Далее вектор w_n засылается в блок I (если $z \in F_1$), или в блок II (если $z \in F_2$).



СПИСОК ЛИТЕРАТУРЫ

1. Розенблат Ф. Принципы нейродинамики. Изд-во «Мир», 1966.
2. Айзерман М. А., Браверман Э. М., Розоноэр Л. И. Теоретические методы потенциальных функций в задаче об обучении автоматов разделению входных ситуаций на классы. «Автоматика и телемеханика», 1964, т. XXIV, № 6.
3. Якубович В. А. Некоторые общие теоретические принципы построения обучаемых опознающих систем. В сб. «Вычислительная техника и вопросы программирования». Изд. ВЦ ЛГУ, 1965, вып. 4.
4. Нильсон Н. Узнающие машины. Изд-во «Мир», 1967.
5. Козинец Б. Н. Об одном алгоритме обучения линейного перцептрона. В сб. «Вычислительная техника и вопросы программирования». Изд. ВЦ ЛГУ, 1964, вып. 3.

В. Н. Вапник, А. А. Журавель, А. Я. Червоненкис

Алгоритмы обучения машин распознаванию образов ОП-1, ОП-2, ОП-3

1. ВВЕДЕНИЕ

Ниже будут описаны три алгоритма обучения машин распознаванию образов, использующие метод «обобщенного портрета»,

С помощью этих алгоритмов строится линейная или кусочно-линейная функция, разделяющая два множества: X и \bar{X} . При этом требуется, чтобы среди векторов обучающей последовательности не было двух тождественных векторов, принадлежащих различным классам.

Чаще при решении практических задач ставится более сильное требование:

$$\frac{\min_{x \in X} |x - \bar{x}|}{\max_{\bar{x} \in \bar{X}} |x - \bar{x}|} > P, \quad (1)$$

где P — некоторое число.

В случае, если неравенство (1) не выполняется, из обучающей последовательности исключаются векторы, на которых достигается $\min_{x, \bar{x}} |x - \bar{x}|$. Обычно задается

$$P \approx 0,05 \div 0,10.$$

Все три алгоритма таковы, что один является составной частью другого, т. е. алгоритм ОП-2 включает в себя ОП-1, а алгоритм ОП-3 в свою очередь — ОП-2. Поэтому все алгоритмы могут быть реализованы в виде одной программы, где нужный алгоритм решения задачи выбирается автоматически.

Время решения существенно зависит от задачи; на машине М-220 эта величина колеблется от нескольких минут до 0,5 ч.

Общим для всех рассматриваемых алгоритмов является метод построения разделяющей гиперплоскости (метод обобщенного портрета [1]).

Особенность метода состоит в следующем: пусть конечные множества векторов $\bar{X}(\bar{x}_1, \dots, \bar{x}_a)$ и $X(x_1, \dots, x_b)$ разделимы гиперплоскостью. Это означает, что существуют такое число k и такой вектор ψ , что:

$$\begin{aligned} (x_1\psi) &\geq 1, \\ &\dots \dots \dots \\ (x_a\psi) &\geq 1, \\ (\bar{x}_1\psi) &\leq k, \\ &\dots \dots \dots \\ (\bar{x}_b\psi) &\leq k, \text{ где } k < 1. \end{aligned} \quad (2)$$

Обобщенным портретом называется минимальный по модулю вектор ψ_0 , удовлетворяющий неравенствам (2). Этот вектор можно представить в виде

$$\psi_0 = \sum_{i=1}^a \alpha_i x_i - \sum_{j=1}^b \beta_j \bar{x}_j,$$

причем

$$\begin{aligned} \alpha_i ((x_i \psi) - 1) &= 0, \quad i = 1, 2, \dots, a; \\ \beta_j ((\bar{x}_j \psi) - k) &= 0, \quad j = 1, 2, \dots, b; \\ \alpha_i &\geq 0; \quad \beta_j \geq 0. \end{aligned} \quad (3)$$

Условие (3) означает, что вектор ψ_0 раскладывается по тем векторам x_i, \bar{x}_j , на которых достигаются равенства: $(x_i \psi) = 1$; $(\bar{x}_j \psi) = k$. Такие векторы мы в дальнейшем будем называть *крайними*, или *информативными*.

Геометрически «обобщенный портрет» (вектор ψ_0) определяет такое направление, проекции множеств X и \bar{X} на которое максимально отстоят друг от друга при условии, что

$$[\max_{\bar{X}} (\bar{x} \psi)] / k \leq \min_X (x \psi). \quad (4)$$

Обычно, если множества X и \bar{X} разделимы гиперплоскостью, то число крайних векторов невелико, т. е. вектор ψ_0 можно представить как линейную комбинацию небольшого числа векторов x_i и \bar{x}_j .

Метод «обобщенного портрета» заключается в выявлении крайних векторов и определении коэффициентов разложения по ним вектора ψ_0 .

По крайним векторам и найденным коэффициентам α_i и β_j легко строится «обобщенный портрет».

Если обучающая последовательность есть случайная выборка длины L из множества $X \cup \bar{X}$, а число крайних векторов равно m , то величина $(1 - m/L)$ характеризует вероятность того, что произвольно выбранный вектор $x \in X \cup \bar{X}$ удовлетворяет одному из неравенств:

$$(x \psi) \geq 1, \text{ если } x \in X,$$

или

$$(x \psi) \leq k, \text{ если } x \in \bar{X}.$$

Таким образом, чем меньше отношение m/L , тем выше вероятность правильной классификации векторов. Можно показать, что в гиперпространстве размерности n

«обобщенный портрет» может быть разложен не более чем по n крайним векторам. Поэтому, если гиперплоскость разделяет обучающую последовательность, численно равную 5—6 размерностям гиперпространства, то она с большой вероятностью будет правильно классифицировать векторы множества $X \cup \bar{X}$.

Практически, достаточно хорошие результаты дает гиперплоскость, построенная по обучающей последовательности, равной $2 \div 3$ размерностям гиперпространства.

Алгоритм ОП-1 реализует метод «обобщенного портрета» (построения разделяющей гиперплоскости).

В случае, если разделяющая гиперплоскость не может быть построена (т. е. если с помощью гиперплоскости нельзя разделить обучающую последовательность), следует воспользоваться алгоритмами ОП-2 или ОП-3.

Алгоритм ОП-2 близок к алгоритму ОП-1. Допустим, что обучающая последовательность не может быть разделена гиперплоскостью (этот факт устанавливается алгоритмом ОП-1), но удалось найти такие r векторов обучающей последовательности, исключив которые можно построить разделяющую гиперплоскость.

В соответствие такой гиперплоскости может быть поставлено число $1 - (r + m)/L$, характеризующее вероятность правильной классификации с помощью этой гиперплоскости (m — число крайних векторов, r — число исключенных векторов, L — общее число векторов обучающей последовательности). Идея алгоритма ОП-2 заключается в минимизации числа $(r + m)/L$.

Однако найти гиперплоскость, на которой число $(r + m)/L$ достигает минимума, задача весьма трудная. Она сводится к отысканию абсолютного экстремума многоэкстремальной функции. Поэтому для поиска экстремума предлагается эвристический алгоритм ОП-2. Теоретически можно гарантировать только, что этот алгоритм отыщет некоторый локальный минимум. Как алгоритм ОП-1, так и ОП-2 принципиально рассчитан на случай, когда среди линейных решающих правил есть удовлетворительное. Практически этими алгоритмами имеет смысл пользоваться, когда $(r + m)/L < 0,3$.

Алгоритм ОП-3 используется в том случае, если среди линейных решающих правил нет удовлетворительного.

Суть этого алгоритма заключается в том, чтобы разделить обучающую последовательность с помощью нескольких гиперплоскостей. При этом лучше экстраполирует то решающее правило, которое состоит из минимального числа гиперплоскостей.

Используя алгоритм ОП-3 в пространстве бинарных векторов, всегда можно разделить любую обучающую последовательность. Однако этот алгоритм не гарантирует, что разделение будет проведено с помощью минимального числа гиперплоскостей. (Отыскание же минимального числа гиперплоскостей сводится к задаче поиска абсолютного экстремума многоэкстремальной функции и потому не имеет удовлетворительного конструктивного решения.)

Следует иметь в виду, что при заданной длине обучающей последовательности наилучшая экстраполяция достигается в том случае, когда задачу удастся решить с помощью алгоритма ОП-1.

Если задачу нельзя решить с помощью алгоритма ОП-1 (не существует гиперплоскости, разделяющей векторы обучающей последовательности), то задачу следует решать при помощи алгоритма ОП-2. При этом желательно увеличить длину обучающей последовательности. Если же и применение алгоритма ОП-2 также не дает нужного результата, то следует воспользоваться алгоритмом ОП-3, вновь увеличив длину обучающей последовательности.

2. ОПИСАНИЕ АЛГОРИТМА ОП-1

Пусть заданы два множества X и \bar{X} . С помощью алгоритма ОП-1 строится минимальный по модулю вектор ψ , удовлетворяющий неравенствам (2):

$$\begin{aligned}(x_i\psi) &\geq 1, & x_i &\in X & (i=1, 2, \dots, a), \\ (\bar{x}_j\psi) &\leq k, & \bar{x}_j &\in \bar{X} & (j=1, 2, \dots, b),\end{aligned}$$

где $k < 1$ — заданное число.

Построенная по найденному вектору ψ гиперплоскость $(x\psi) = (1+k)/2$ разделяет множества X и \bar{X} ; при выполнении условия (4) эта гиперплоскость максимально от них удалена. С помощью алгоритма ОП-1 устанавливается (в случае, когда это так), что не существует гиперплоскости, разделяющей векторы обучающей последовательности и отстоящей от ближайшего из век-

торов обучающей последовательности на величину, большую ρ , где ρ — допустимое расстояние (параметр алгоритма).

Так как вектор ψ можно представить в виде

$$\psi = \sum_{i=1}^a \alpha_i x_i - \sum_{j=1}^b \beta_j \bar{x}_j,$$

где $\alpha_i \geq 0$, $\beta_j \geq 0$, то для того чтобы его найти, достаточно отыскать соответствующие значения α_i и β_j .

Известно [1], что α_i и β_j есть координаты точки покоя устойчивой системы уравнений:

$$\begin{aligned} \dot{\alpha}_i &= \begin{cases} 1 - (x_i \psi), & \text{если } 1 - (x_i \psi) \geq 0 \text{ или } \alpha_i > 0, \\ 0 & \text{в противном случае;} \end{cases} \\ \dot{\beta}_j &= \begin{cases} -k + (\bar{x}_j \psi), & \text{если } -k + (\bar{x}_j \psi) \geq 0 \text{ или } \beta_j > 0, \\ 0 & \text{в противном случае.} \end{cases} \end{aligned} \quad (5)$$

Здесь $i = 1, \dots, a$; $j = 1, \dots, b$.

Суть алгоритма ОП-1 заключается в отыскании точки покоя системы (5).

Заметим, что каждому вектору из обучающей последовательности соответствует свое уравнение в системе (5). Иногда система может оказаться столь громоздкой, что решать ее целиком на ЦВМ неудобно, поэтому она решается по частям: обучающая последовательность разбивается на μ групп и сначала решается система уравнений, соответствующая первой группе векторов; далее определяются крайние векторы из первой группы, т. е. такие векторы, которым в точке покоя системы соответствуют значения α_i , $\beta_j \neq 0$.

По этим векторам строится вектор ψ_1 , с помощью которого к крайним векторам первой группы присоединяются некоторые векторы второй группы по правилу: вектор x , принадлежащий множеству X , присоединяется к крайним, если

$$(x \psi_1) < 1 - \delta. \quad (6)$$

Если же вектор принадлежит множеству \bar{X} , то он присоединяется к крайним в случае, когда

$$(\bar{x} \psi_1) > k + \delta, \quad (6')$$

где δ — некоторая константа.

Затем составляется следующая система, отыскиваются новые крайние векторы, определяется вектор ψ_2 и

к крайним векторам присоединяются векторы из первых трех групп, удовлетворяющие неравенствам вида (6) и (6'), и т. д., вплоть до того момента, пока во всех m группах не останется ни одного вектора, удовлетворяющего этим неравенствам.

Смысл такого решения заключается в том, что вначале находятся крайние векторы, а затем определяется разложение по ним вектора ψ . Этот способ наиболее целесообразен, так как число крайних векторов, как правило, невелико. Напомним, что геометрически крайние векторы множества X — это те векторы, которых впервые коснется разделяющая гиперплоскость при ее параллельном переносе в направлении множества X . Аналогично определяются крайние векторы множества \bar{X} .

Одновременно с вычислением вектора ψ проверяется существование разделяющей гиперплоскости. С этой целью при движении α, β к точке покоя в каждой точке траектории вычисляется величина

$$W(\alpha, \beta) = \sum_{i=1}^a \alpha_i - k \sum_{j=1}^b \beta_j - \frac{1}{2} (\dot{\psi}, \psi),$$

где

$$\psi = \sum_{i=1}^a \alpha_i x_i - \sum_{j=1}^b \beta_j \bar{x}_j.$$

Величина $W(\alpha, \beta)$ при этом возрастает. Оказывается, что если не существует разделяющей гиперплоскости, то в какой-то момент времени выполнится неравенство

$$W(\alpha, \beta) \geq \frac{(1-k)^2}{2\rho^2}. \quad (7)$$

Это означает, что не существует разделяющей гиперплоскости, проходящей от ближайшего из векторов обучающей последовательности на расстоянии, большем ρ .

Таким образом, с помощью алгоритма ОП-1 либо строится разделяющая гиперплоскость, либо устанавливается, что ее не существует.

Искать точку покоя системы (5) удобно с помощью модифицированного метода параллельных касательных — «партан», который предложен для поиска экстремума отрицательно определенных квадратичных форм [2],

С помощью этого метода удается найти экстремум за $(2p-1)$ шагов, где p — ранг формы.

Поиск точки покоя системы (5) эквивалентен отысканию в положительном квадранте ($\alpha_i \geq 0$, $\beta_j \geq 0$, $i=1, 2, \dots, a$; $j=1, 2, \dots, b$) максимума квадратичной формы:

$$W(\alpha, \beta) = \sum_{i=1}^a \alpha_i - k \sum_{j=1}^b \beta_j - 1/2 (\psi\psi), \quad (8)$$

где

$$\psi = \sum_{i=1}^a \alpha_i x_i - \sum_{j=1}^b \beta_j \bar{x}_j.$$

Для отрицательно определенной формы (8) максимум существует и поэтому существует точка покоя системы (5), а следовательно, и вектор ψ , удовлетворяющий условиям (2).

Если же максимума формы (8) не существует (т. е. функция $W(\alpha, \beta)$ в положительном квадранте может быть сколь угодно велика), то не существует вектора ψ , удовлетворяющего условию (2). Поиск максимума формы (8) не во всем пространстве α, β , а только в части его, естественно, затрудняет решение задачи (по существу, решается задача квадратичного программирования, правда, при достаточно простых ограничениях: $\alpha_i \geq 0$). В этих условиях максимум определяется не за $(2p-1)$, а, вообще говоря, за большее число шагов.

Геометрически (на плоскости) идея метода «партан» заключается в следующем. Пусть $f(x)=0$ — эллипс, а P_1 и P_2 — любые две точки на плоскости X . Из точки P_2 будем двигаться параллельно касательной к точке P_1 , пока не достигнем минимума $f(x)$ в точке P_3 . Так как в точках P_1 и P_3 касательные параллельны, то абсолютный минимум находится в экстремальной точке P_4 на прямой, проходящей через P_1 и P_3 (рис. 1).

Алгоритм «партан» таков, что в p -мерном случае в точках P_{2p-4} и P_{2p-1} касательные плоскости параллельны, и экстремум функции $f(x)$ находится на прямой, соединяющей P_{2p-4} и P_{2p-1} , т. е. в точке P_{2p} . Пусть P_0, P_2, \dots, P_{2p} — последовательность точек в p -мерном пространстве (первую точку для удобства обозначим не P_1 , а P_0).

Из точки P_0 движемся вдоль ломаной (рис. 2), у которой P_i — точка максимума на прямой, соединяющей эту точку с предшествующей.

Направления отрезков $P_2P_0, P_3P_2, \dots, P_{2k+1}P_{2k}, \dots, P_{2p-1}, P_{2p-2}$ определяют соответственно $\text{grad } f(P_0), \text{grad } f(P_2), \dots, \text{grad } f(P_{2k}), \dots$. Направления же отрезков

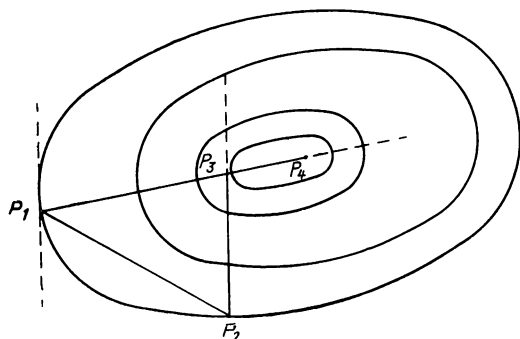


Рис. 1.

$P_4P_3, \dots, P_{2k-1}P_{2k-4}, \dots, P_{2p}P_{2p-1}$ определяются разностями $P_3-P_0, \dots, P_{2k-1}P_{2k-4}$. Движение вдоль отрезков $P_{2k+1}P_{2k}$ будем называть градиентным шагом, а вдоль отрезков $P_{2k}P_{2k-1}$ — «овражным» шагом.

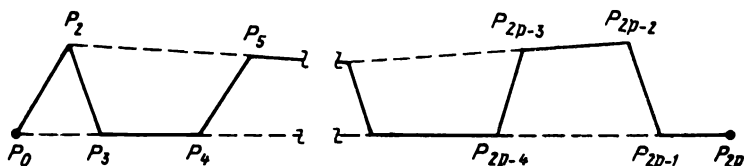


Рис. 2.

Движение вдоль любого отрезка происходит вплоть до достижения экстремума функции $f(x)$.

Если, например, отрицательно определенная форма задана в виде

$$W(\alpha\beta) = \sum \alpha_i - k \sum \beta_j - 1/2(\psi),$$

где $\psi = \sum \alpha_i x_i - \sum \beta_j \bar{x}_j$, то за начальную точку выбирается любая точка пространства α, β , а направление движения — вдоль градиентного шага:

$$\text{grad } W(\alpha, \beta) = (1 - x_1, \psi); \dots, (-k + \bar{x}_b \psi).$$

Обозначим через $\dot{\alpha}_1, \dots, \dot{\alpha}_a, \dot{\beta}_1, \dots, \dot{\beta}_b$ координаты $\text{grad } W(\alpha\beta)$. Тогда точка максимума $W(\alpha\beta)$ при движении по градиентному шагу будет достигнута при величине шага

$$h = [\Sigma \dot{\alpha}_i - k \Sigma \dot{\beta}_j - (\psi\dot{\psi})] / \|\dot{\psi}\|^2,$$

где

$$\dot{\psi} = \sum_{i=1}^a \dot{\alpha}_i x_i - \sum_{j=1}^b \dot{\beta}_j x_j.$$

Координаты этой точки могут быть вычислены:

$$\alpha(2k+1) = \alpha(2k) + \text{grad } W[\alpha(2k); \beta(2k)]h,$$

$$\alpha(2k+1) = \beta(2k) + \text{grad } W[\alpha(2k); \beta(2k)]h.$$

Направление движения по «овражному» шагу вычисляется как

$$\Delta^\alpha = \alpha(2k-1) - \alpha(2k-4),$$

$$\Delta^\beta = \beta(2k-1) - \beta(2k-4),$$

а величина шага — по формуле

$$h = [\Sigma \Delta_i^\alpha - k \Sigma \Delta_j^\beta - (\psi\psi_\Delta)] / \|\psi_\Delta\|^2,$$

где

$$\psi_\Delta = \Sigma \Delta_i^\alpha x_i - \Sigma \Delta_j^\beta x_j.$$

Координаты точки $\alpha(2k)$ $\beta(2k)$ вычисляются так:

$$\alpha(2k) = \alpha(2k-1) + \Delta^\alpha h,$$

$$\beta(2k) = \beta(2k-1) + \Delta^\beta h.$$

Когда максимум отрицательно определенной формы $W(\alpha\beta)$ определяется в положительном квадранте, направление градиентного шага задается системой (5). При выборе величины шага надо следить за тем, чтобы следующая точка траектории движения не вышла за пределы положительного квадранта. Для этого величина градиентного шага определяется как

$$h^* = \min [|\alpha/\dot{\alpha}|, \dots, |\beta/\dot{\beta}|, h],$$

где минимум берется по тем отношениям $|\alpha/\dot{\alpha}|$, $|\beta/\dot{\beta}|$, для которых величина $\dot{\alpha}$, $\dot{\beta}$ отрицательна. При выборе вели-

чины „овражного“ шага аналогично определяется величина

$$h^* = \min [|\alpha/\Delta^\alpha|, \dots, |\beta/\Delta^\beta|, h],$$

Система (5) решается до тех пор, пока не выполняются либо неравенства

$$|\dot{\alpha}_1| \leq \varepsilon \dots |\dot{\alpha}_a| \leq \varepsilon; |\dot{\beta}_1| \leq \varepsilon \dots |\dot{\beta}_b| \leq \varepsilon, \quad (9)$$

либо неравенство

$$W(\alpha, \beta) > (1-k)^2/2\rho^2. \quad (10)$$

Выполнение неравенств (9) означает, что разделяющая гиперплоскость построена. Выполнение же неравенства (10) показывает, что такая гиперплоскость построена быть не может.

3. ОПИСАНИЕ ПРОГРАММЫ ОП-1

Программа может быть написана в двух вариантах:

- 1) для случая, когда векторы бинарны;
- 2) для случая, когда координаты векторов могут принимать значения: $a \leq x \leq b$.

Разница между этими вариантами заключается в способе размещения в памяти машины векторов обучающей последовательности и в способе вычисления скалярного произведения.

Когда векторы бинарны, каждая координата записывается в одном разряде ячейки памяти. (Если количество координат вектора больше числа разрядов ячейки, вектор может занимать несколько ячеек.)

Первая ячейка группы ячеек, в которых записан вектор, содержит один разряд для индексации (если в этом разряде записан «0», то вектор относится к первому классу, если 1 — то ко второму). В этой ячейке первые девять разрядов удобно использовать не для записи вектора, а заносить туда номера векторов. Восьмеричные числа 0—377 — номера векторов первого класса, а 400—777 — номера векторов второго класса. При такой нумерации первый разряд одновременно используется как индекс класса.

Обычно используется программа, рассчитанная на бинарные векторы. Для решения конкретной задачи состав-

ляется информационная карта, состоящая из десятичных констант, которые означают соответственно:

- 1) размерность пространства n ;
- 2) количество ячеек, занимаемых одним вектором z ;
- 3) величину параметра алгоритма k ;
- 4) величину ε , определяющую точность вычисления (задача считается решенной при выполнении неравенств $(x_i\psi) \geq 1 - \varepsilon$, $(\bar{x}_j\psi) \leq k + \varepsilon$);
- 5) величину константы «неделения» $W_0 = (1 - k)^2 / \rho^2$;
- 6) точность «внутреннего экзамена» δ : при поиске крайних векторов отбираются векторы x , \bar{x} , для которых $(x\psi) < 1 - \delta$; $(\bar{x}\psi) > k + \delta$;
- 7) количество векторов в группе M .

Ниже описана блок-схема программы ОП-1.

Блок ОП-1-1

1. Переводит константы из десятичных в двоичные.
2. Передает управление блоку ОП-1-2

Блок ОП-1-2

1. Засылает векторы на магнитный барабан МБ-2 (блок нужен только для программы ОП-3).
2. Передает управление блоку ОП-1-3.

Блок ОП-1-3

1. Подсчитывает количество векторов в обучающей последовательности, разбивает их на группы по M векторов в каждой (последняя группа может быть неполной).
2. Засылает группы на МБ-3.
3. Передает управление блоку ОП-1-4.

Блок ОП-1-4

1. Записывает нули в группу рабочих ячеек (группы рабочих ячеек описаны ниже).
2. Передает управление блоку ОП-1-5.

Блок ОП-1-5

1. Отсылает на место информативных векторов первую группу векторов.
2. Передает управление блоку ОП-1-6.

Блок ОП-1-6 («обучение»)

1. Решает систему (5), ниже этот блок будет подробно рассмотрен.
2. Передает управление блоку ОП-1-7.

Блок ОП-1-7

1. Сокращает массив информативных векторов, исключая из группы информативных векторов те векторы, которые входят в раз-

ложение ψ с нулевыми весовыми коэффициентами, т. е. векторы, которым в группе ячеек R_{α}^I соответствуют нули (см. блок «обучение»).

2. Пододвигает (записывает подряд) информативные векторы и соответствующие им коэффициенты разложения α , β .

3. Передает управление блоку ОП-1-8.

Блок ОП-1-8

1. Добавляет к оставшимся информативным векторам новую группу информативных векторов.

Для этого:

а) если программа использовала этот блок i раз ($i < \mu$, где μ — число групп), то информативные векторы отбираются среди первых $(i+1)$ групп, которые поочередно вызываются в магнитное оперативное запоминающее устройство (МОЗУ). Каждый вектор группы проверяется на выполнение условия

$$(x\psi) < 1 - \delta, \text{ если } x \in X; \quad (11)$$

$$(\bar{x}\psi) > k + \delta, \text{ если } \bar{x} \in \bar{X}.$$

Если эти условия выполняются, то вектор x (\bar{x}) добавляется к группе информативных векторов, а в соответствующей ему ячейке группы R_{α}^I записывается «0»;

б) если программа выходила на этот блок i раз ($i \geq \mu$), то информативные векторы отбираются из всех групп.

2. Подсчитывает число добавленных информативных векторов:

а) если число добавленных информативных векторов равно нулю, то управление передается блоку ОП-1-9;

б) если же число добавленных информативных векторов отлично от нуля, то определяет общее число информативных векторов и к содержимому счетчика $Sч1$ добавляет единицу ($Sч1$ подсчитывает, сколько раз программа использовала этот блок). Затем передает управление блоку ОП-1-6.

Блок ОП-1-9

1. Печатает вектор ψ .

2. Останов.

Блок ОП-1-10

1. Печатает число 777000777 (информация о неделении).

2. Останов.

Подробная блок-схема блока «обучение»

Блок «обучение» использует группы рабочих ячеек: R_{α}^I , R_{α}^{II} , R_{α}^{III} , R_{α}^I , R_{ψ}^I , R_{ψ}^{II} , R_{ψ}^{III} , R_{ψ}^{\bullet} .

(При решении системы (5) методом «партан» надо постоянно хранить в памяти три точки траектории.)

Блок ОП-1-6-1 (ψ — вне цикла)

1. Вычисляет ψ по информативным векторам $x_i(\bar{x}_j)$ и коэффициентам α, β , записанным в группе ячеек R_α^I (в начальный момент $\alpha = \beta = 0$ — начальные условия), т. е. вычисляет вектор

$$\psi = \sum_{i=1}^a \alpha_i x_i - \sum_{j=1}^b \beta_j \bar{x}_j.$$

Вектор ψ записывается в группу ячеек R_ψ^I .

2. Передает управление блоку ОП-1-6-2.

Блок ОП-1-6-2 (первая пересылка вне цикла)

1. Переписывает содержимое группы ячеек R_α^I в группу $R_\alpha^{II} \times (R_\alpha^I \rightarrow R_\alpha^{II})$. Содержимое группы ячеек R_ψ^I — в R_ψ^{II} .

2. Передает управление блоку ОП-1-6-3.

Блок ОП-1-6-3 (градиентный шаг вне цикла)

1. Вычисляет производные $\dot{\alpha}_i(\dot{\beta}_j)$ и записывает их в группу R_α . Вычисление проводится по формулам:

$$\dot{\alpha}_i = \begin{cases} 1 - (x_i \psi), & \text{если } 1 - (x_i \psi) \geq 0 \text{ или } \alpha_i \geq 0, \\ 0 & \text{в противном случае;} \end{cases}$$

.....

$$\dot{\beta}_j = \begin{cases} -k + (\bar{x}_j \psi), & \text{если } -k + (\bar{x}_j \psi) \geq 0 \text{ или } \beta_j \geq 0, \\ 0 & \text{в противном случае,} \end{cases}$$

Здесь ψ — вектор, записанный в группе ячеек R_ψ^I .

2. Вычисляет вектор $\dot{\psi} = \sum \dot{\alpha}_i x_i - \sum \dot{\beta}_j \bar{x}_j$ (величины $\dot{\alpha}_i, \dot{\beta}_j$ и записываются в группе ячеек R_α , а $\dot{\psi}$ — в группе ячеек R_ψ).

3. Вычисляет величину шага. Для этого:

а) вычисляет числитель пробного шага

$$\psi = \sum_{i=1}^a \dot{\alpha}_i - k \sum_{j=1}^b \dot{\beta}_j - (\dot{\psi} \dot{\psi}),$$

где величины $\dot{\alpha}_i, \dot{\beta}_j$ хранятся в группе ячеек R_α , $\dot{\psi}$ — в R_ψ^I , а $\dot{\psi}$ — в R_ψ ;

б) вычисляет знаменатель пробного шага $3n = (\dot{\psi} \dot{\psi})$;

в) вычисляет величину пробного шага $h_{пр} = \psi / 3n$;

г) вычисляет величину шага

$$h = \min [h_{пр}, |\alpha_i / \dot{\alpha}_i|; |\beta_j / \dot{\beta}_j|],$$

отношения $\alpha_i/\dot{\alpha}_i$, $\beta_j/\dot{\beta}_j$ берутся по тем i, j , для которых $\dot{\alpha}_i < 0$, $\dot{\beta}_j < 0$.

4. Вычисляет новое приближение вектора ψ_n : $\psi_n = \psi + \dot{\psi}h$, вектор ψ хранится в группе ячеек R_ψ^I , ψ_n записывается в группу R_ψ^I .

5. Вычисляется новое приближение α_n , β_n : $\alpha_n = \alpha + \dot{\alpha}h$, $\beta_n = \beta + \dot{\beta}h$, величины α , β хранятся в группе ячеек R_α^I , α_n , β_n записывается в группу R_α^I .

6. Передает управление блоку ОП-1-6-4.

Блок ОП-1-6-4 (первая пересылка в цикле).

1. Переписывает содержимое ячеек R_α^{II} в группы ячеек R_α^{III} ($R_\alpha^{II} \rightarrow R_\alpha^{III}$).

2. $R_\alpha^I \rightarrow R_\alpha^{II}$.

3. $R_\psi^{II} \rightarrow R_\psi^{III}$.

4. $R_\psi^I \rightarrow R_\psi^{II}$.

5. Передает управление блоку ОП-1-6-5

Блок ОП-1-6-5 (градиентный шаг в цикле)

1. Выполняет операции п. 1 блока ОП-1-6-3.

2. Проверяет выполнение условий (9).

Если эти условия выполнены, то передает управление блоку ОП-1-7.

Если же хотя бы для одного $\dot{\alpha}_i$, $\dot{\beta}$ неравенство не выполняется, то передает управление блоку ОП-1-6-5 (п. 3).

3. Выполняет операции пп. 2, 3, 4, 5 блока ОП-1-6-3.

4. Передает управление блоку ОП-1-6-6.

Блок ОП-1-6-6 (проверка условия неделения)

1. Вычисляет величину

$$W(\alpha, \beta) = \sum \alpha_i - k \sum \beta_j - 1/2(\psi\psi).$$

Здесь величины α_i , β_j — из группы R_α^I ; ψ — из группы R_ψ^I .

2. Проверяет выполнение неравенства

$$W(\alpha, \beta) \geq W_0 = (1-k)^2/2\rho^2. \quad (10)$$

Если это неравенство выполнено, то передает управление блоку ОП-1-10. Если же $W(\alpha, \beta) < W_0$, то передает управление блоку ОП-1-6-7.

Блок ОП-1-6-7 (овражный шаг)

1. Вычисляет разности $\Delta\alpha = \alpha^I - \alpha^{III}$, $\Delta\beta = \beta^I - \beta^{III}$.

Здесь величины α^I , β^I — из группы R_α^I ; α^{III} , β^{III} — из группы R_α^{III} ; разность $\Delta\alpha$, $\Delta\beta$ записывается в группу ячеек R_α^I .

2. Вычисляет разности

$$\Delta\psi = \psi^I - \psi^{III}.$$

где ψ^I — из группы R_ψ^I ; ψ^{III} — из группы R_ψ^{III} ; разность $\Delta\psi$ записывает в группу ячеек R_ψ .

3. Выполняет операции пп. 3, 4, 5 блока ОП-1-6-3.

4. Передает управление блоку ОП-1-6-4.

4. ОПИСАНИЕ АЛГОРИТМА ОП-2

Программа ОП-2 применяется в том случае, когда нельзя разделить векторы обучающей последовательности с помощью гиперплоскости (в этом случае в программе ОП-1 произойдет «останов», так как выполнится неравенство $W(\alpha\beta) \geq W_0$). Эта программа исключает некоторое число векторов, препятствующих разделению гиперплоскостью векторов обучающей последовательности.

Исключение векторов производится по следующему правилу. Пусть в некоторый момент окажется, что разделение невозможно, т. е.

$$W(\alpha, \beta) = \sum \alpha_i - k \sum \beta_j - 1/2(\psi\psi) \geq W_0.$$

Заметим, что исключение из числа информативных векторов вектора x , \bar{x} с соответствующими значениями α^* , β^* уменьшает значение функции $W(\alpha, \beta)$ на величину

$$\sigma^* = \alpha^* \alpha^* + \alpha^{*2} \|x\|^2 / 2, \quad \sigma^* = \beta^* \beta^* + \beta^{*2} \|\bar{x}\|^2 / 2.$$

Из множества информативных векторов исключается тот вектор, которому соответствует максимальная величина σ^* . Оставшееся множество векторов вновь делится с помощью гиперплоскости. Если разделение опять невозможно, то из обучающей последовательности исключается еще один вектор, и т. д. до тех пор, пока оставшееся множество векторов не будет разделено с помощью гиперплоскости.

5. ОПИСАНИЕ ПРОГРАММЫ ОП-2

Программа ОП-2 включает в себя как составную часть программу ОП-1 (со следующим изменением: в блоке ОП-1-10 вместо операции «Останов» выполняется операция «Передать управление блоку ОП-2-1»).

Информационная карта та же, что и к программе ОП-1.

Рассмотрим описание блок-схемы программы ОП-2.

Блок ОП-2-1 (поиск максимума)

1. Отыскивает максимальное значение σ .
2. Запоминает порядковый номер i , при котором величина σ максимальна.
3. Передает управление блоку ОП-2-2.

Блок ОП-2-2 (поиск вектора)

1. По порядковому номеру i устанавливает номер N i -го информативного вектора.
2. Передает управление блоку ОП-2-3.

Блок ОП-2-3

1. Исключает из информативных векторов вектор с номером N .
2. Печатает номер исключенного вектора.
3. Пододвигает (записывает подряд) оставшиеся информативные векторы.
4. Пододвигает соответствующие значения α , β (в группе ячеек R_{α}^I).
5. Передает управление блоку ОП-2-4.

Блок ОП-2-4 (исключение векторов)

1. Вызывает последовательно по группам с МБ-2 векторы обучающей последовательности, находит и исключает среди этих векторов вектор с номером N . Пододвигает векторы и образует новые группы по M векторов в каждой (последняя группа может быть неполной), записывает эти группы на МБ-2.
2. Передает управление блоку ОП-2-5.

Блок ОП-2-5 (счетчик)

1. Добавляет к содержимому счетчика числа исключенных векторов $S_{ч2}$ единицу.
2. Передает управление блоку ОП-1-6.

6. ОПИСАНИЕ АЛГОРИТМА ОП-3

Когда число векторов, исключенных с помощью программы ОП-2, сравнительно велико, что свидетельствует о недостаточно высоком качестве найденного решающего правила, используется программа ОП-3: делается попытка разделить векторы обучающей последовательности гиперповерхностью, составленной из двух гиперплоскостей: за одну принимается гиперплоскость, построенная с помощью программы ОП-2, другая же строится так, чтобы составляемая поверхность правильно разделяла как можно больше векторов обучающей последовательности.

Если с помощью этой гиперповерхности не удастся разделить все векторы обучающей последовательности,

то строится гиперповерхность, состоящая из трех гиперплоскостей и т. д.

Если векторы обучающей последовательности бинарны, то с увеличением числа гиперплоскостей, количество неправильно опознанных векторов уменьшается. Поэтому разделить обучающую последовательность всегда удается (если нет двух тождественных векторов, принадлежащих разным классам).

Однако при фиксированной длине обучающей последовательности чем меньше число гиперплоскостей, составляющих гиперповерхность, тем выше качество решающего правила.

Итак, пусть обучающая последовательность состоит из L векторов размерности n и пусть $(x\psi) = (1+k)/2$ — гиперплоскость, правильно разделяющая L_1 векторов. С помощью этой гиперплоскости из L векторов размерности n образуем новый массив из L векторов размерности $(n+2)$ по следующему правилу: каждому вектору $x(\bar{x})$ из обучающей последовательности ставится в соответствие вектор $x'(\bar{x}')$ в пространстве размерности $(n+2)$, у которого первые n координат совпадают с координатами вектора $x(\bar{x})$, а $(n+1)$ -я и $(n+2)$ -я координаты есть соответственно:

$$\langle 1 \rangle, \langle 0 \rangle, \text{ если } (x\psi) \geq (1+k)/2; ((\bar{x}\psi) > (1+k)/2)$$

или

$$\langle 0 \rangle, \langle 1 \rangle, \text{ если } x\psi < (1+k)/2; ((\bar{x}\psi) < (1+k)/2).$$

Полученное множество векторов разделим на два класса по правилу: вектор x' относится к первому классу, если соответствующий ему вектор x относится к первому классу, и вектор \bar{x}' относится ко второму классу, если вектор \bar{x} принадлежит второму классу:

Разделим теперь множество векторов $\bar{X}' \cup X'$ на два класса с помощью гиперплоскости $(x'\psi') = (1+k)/2$. В пространстве размерности n такому разделению соответствует разделение векторов $X \cup \bar{X}$ с помощью двух гиперплоскостей. Если же и с помощью двух гиперплоскостей разделение невозможно, то обучающая последовательность аналогично разделяется с помощью трех гиперплоскостей и т. д.

7. ОПИСАНИЕ ПРОГРАММЫ ОП-3

Программа ОП-3 включает в себя как составную часть программу ОП-2. Для работы с программой ОП-3 в информационную карту добавляется константа P .

Программа будет строить новую гиперплоскость, если окажется, что $(m^*+r)/L > P$, где m^* — число информативных векторов для гиперплоскости, построенной с помощью программы ОП-2, r — число векторов, исключенных при построении гиперплоскости с помощью программы ОП-2.

Рассмотрим описание блок-схемы программы ОП-3.

Блок ОП-2

В программе ОП-2 внесены изменения: в блоке ОП-1-9 вместо операции «Останов» выполняется операция «Передать управление блоку ОП-3-1».

Блок ОП-3-1

1. Вычисляет величину $(m^*+r)/L$ и сравнивает ее с эталоном P .

2. Останов, если $(m^*+r)/L \leq P$; если это неравенство не выполнено, то записывает «0» в ячейку R_p и передает управление блоку ОП-3-2.

Блок ОП-3-2 (расширение пространства)

1. Вызывает с МБ-3 векторы обучающей последовательности.

2. Дописывает $(n+1)$ -ю и $(n+2)$ -ю координаты по правилу: если для вектора x , \bar{x} $(x\psi) \geq (1+k)/2$, то дописываются координаты «1», «0», если же $(x\psi) < (1+k)/2$, то дописываются координаты «0», «1». Записывает полученные векторы на МБ-3.

3. Разбивает полученные векторы на группы по M векторов в каждой и записывает эти группы на МБ-2.

4. Передает управление блоку ОП-3-3.

Блок ОП-3-3

1. Записывает вектор ψ на МБ-3.

2. Передает управление блоку ОП-1-2.

8. ПРОГРАММА «ЭКЗАМЕН»

Для классификации векторов, не включенных в обучающую последовательность, может быть написана программа, которая, используя выработанное в процессе обучения решающее правило, классифицировала бы предъявленные векторы.

Для программ ОП-1 и ОП-2 с помощью программы «экзамен» выясняется, к какому классу принадлежит

данный вектор, т. е. по какую сторону от построенной в процессе обучения гиперплоскости он находится. Причем, считается, что вектор x принадлежит первому классу, если $(x\psi) \geq (1+k)/2$, и второму классу, если $(x\psi) < (1+k)/2$.

Пусть решающее правило получено с помощью d гиперплоскостей:

$$\{(x\psi) = (1+k)/2, (x^{(1)}\psi^{(1)}) = (1+k)/2, \dots, \\ \dots, (x^{(d-1)}\psi^{(d-1)}) = (1+k)/2\}.$$

Тогда для классификации с помощью решающего правила, найденного алгоритмом ОП-3, поступают так:

1. $d-1$ раз увеличивают число координат классифицируемого вектора x по правилу: если $(x^s\psi^s) > (1+k)/2$, то дописывается $(s+1)$ -я и $(s+2)$ -я координаты — соответственно, «1», «0»; если это неравенство не выполняется, то соответствующие координаты равны «0» и «1». Процесс повторяется $(d-1)$ раз.

2. Определяется выполнение неравенства $(x^{d-1}\psi^{d-1}) \geq (1+k)/2$. Если это неравенство выполнено, то вектор x принадлежит первому классу, в противном случае — второму классу.

Программы «экзамен» для алгоритмов ОП-1, ОП-2, ОП-3 могут быть написаны в виде одной программы.

9. ПРОГРАММА «СКОЛЬЗЯЩИЙ КОНТРОЛЬ»

В случае, когда нет материала для проведения экзамена, принято использовать «скользящий контроль». Идея этого «экзамена» заключается в следующем: из обучающей последовательности извлекается один вектор, на оставшихся векторах проводят «обучение», а этот вектор «экзаменуют»; затем то же самое проделывают с другим вектором и т. д.

Для программ ОП-1 и ОП-2 при проведении такого «экзамена» нет необходимости проверять все векторы. Заведомо известно, что неинформативные векторы (т. е. векторы, участвовавшие в разложении вектора ψ с нулевыми коэффициентами, будут опознаны правильно. Поэтому достаточно провести «скользящий контроль» только на информативных векторах.

Такая программа использует программу ОП-1 в качестве блока. Опишем ее блок-схему.

Блок 1

1. Записывает на МБ-3 информативные векторы и коэффициенты разложения по ним вектора ψ . (При вводе обучающей последовательности информативные векторы вводятся первыми). Их количество указывается в информационной карте к программе «скользящий контроль». Записывает «0» в счетчик.

2. Передает управление блоку 2.

Блок 2

1. Делит векторы на группы и записывает их на МБ-2.

2. Переписывает информативные векторы и коэффициенты α в группу рабочих ячеек R^I_α .

3. Передает управление блоку 3.

Блок 3

1. Прибавляет к содержимому счетчика числа информативных векторов единицу.

Если содержимое счетчика превосходит количество векторов, то выполняет операцию «Останов». В противном случае передает управление блоку 4.

Блок 4

1. Устанавливает номер i -го информативного вектора (i есть содержимое счетчика числа информативных векторов). Исключает вектор с этим номером из числа информативных и записывает его в рабочие ячейки R^* . Пододвигает информативные векторы и соответствующие им коэффициенты разложения.

3. Исключает вектор с этим номером из обучающей последовательности, записанной на МБ-2. Пододвигает оставшиеся векторы и разбивает их на группы.

4. Передает управление блоку ОП-1-6 программы ОП-1. В режиме «скользящий контроль» в программу ОП-1 внесены изменения: в блоке ОП-1-9 вместо операции «Останов» выполняется операция: «Передать управление блоку 5».

Блок 5

1. Проверяет, по какую сторону от построенной гиперплоскости находится i -й информативный вектор (он хранится в рабочей ячейке R^*).

2. Печатает, к какому классу отнесен вектор.

3. Вызывает в МОЗУ содержимое третьего барабана.

4. Передает управление блоку 2.

СПИСОК ЛИТЕРАТУРЫ

1. В алник В. Н., Червоненкис А. Я. Алгоритмы с полной памятью и рекуррентные алгоритмы в задаче обучения распознаванию образов. «Автоматика и телемеханика», 1968, № 4.

2. Buchler R. J., Shah B. V. and Keinptiore O. The method of parallel tangents (PARTAN) for finding an optimum. Tech. Ref. N2 office of Naval Research Contr. Nom — 5030(05) Jawa state University Statistical Laboratory. April 1961.

Алгоритм обучения распознаванию образов «Кора»

Алгоритм «Кора» предназначен для решения задач распознавания образов, относящихся к одному из L классов. Этот алгоритм строит решающее правило с помощью обучающего множества объектов, принадлежность которых к тому или иному классу известна. Каждый объект этого множества кодируется бинарным вектором x_1, \dots, x_n , размерность которого не более 45.

Описываемый алгоритм основан на поиске для каждого класса таких признаков, которыми обладают некоторые примеры данного класса и не обладает ни один из примеров других классов. Такие признаки будем называть достаточными на множестве примеров. Для этих признаков вводится оценка качества; качество признака определяется числом примеров, которые им обладают.

При весьма общих предположениях вероятность ошибки распознавания при использовании данного признака монотонно убывает с ростом его оценки. Алгоритм «Кора» для каждого класса объектов отбирает набор признаков, обладающий следующими свойствами:

- 1) каждый из признаков этого набора имеет достаточно большую оценку качества;
- 2) дисперсия числа отобранных признаков, которыми обладает каждый из примеров (по обучающему множеству), должна быть возможно меньше.

Отбор признаков производится среди всевозможных конъюнкций переменных x_1, \dots, x_n , описывающих объекты, и их отрицаний. Считается, что объект обладает данным признаком, если конъюнкция, соответствующая этому признаку, принимает на объекте единичное значение. Для сокращения времени перебора максимальная длина перебираемых конъюнкций ограничивается.

При распознавании подсчитывается число отобранных признаков каждого класса, которыми обладает объект, подлежащий распознаванию. Объект относится к тому классу, для которого это число оказывается наибольшим. Рассмотрим более подробно алгоритм обучения, т. е. перебора конъюнкций и отбора среди них достаточных признаков каждого класса.

Перед началом работы задаются максимальной длиной l перебираемых конъюнкций, максимальным числом

m признаков каждого класса, которые должны быть отобраны, и начальными значениями \bar{k}_i^0 минимальных оценок \bar{k}_i качества признаков, отбираемых для класса A_i . В процессе работы алгоритма эти оценки могут изменяться.

Алгоритм обучения условно может быть разбит на четыре основных блока: «анализ», «проверка», «продолжение» и «провал», функции которых описываются ниже.

Блок «анализ» рассматривает произвольную конъюнкцию длиной $r \leq l$:

$$\beta = x_{i_1}^{\sigma_1} \wedge x_{i_2}^{\sigma_2} \wedge \dots \wedge x_{i_r}^{\sigma_r},$$

где

$$x^{\sigma} = \begin{cases} x, & \text{если } \sigma = 0; \\ \bar{x}, & \text{если } \sigma = 1; \end{cases}$$

$1 \leq i_p \leq n$ и $i_p < i_q$ при $p < q$.

Рассмотрение начинается с конъюнкции x_1 . Обозначим через k_j число примеров класса A_j , на которых $\beta = 1$. При этом возможны следующие случаи:

1. Существует такое j , что $k_j \geq \bar{k}_j$ и для всех $i \neq j$ $k_i = 0$. В этом случае конъюнкция β является достаточным на множестве примеров признаком класса A_j с оценкой $k_j \geq \bar{k}_j$. Далее эта конъюнкция поступает на вход блока «проверка», где решается вопрос о том, стоит ли ее заносить в память.

2. Существуют такие i и j , что $k_i \geq \bar{k}_i$ и $k_j \neq 0$. В этом случае β не является достаточным признаком класса A_i , однако ее, возможно, удастся использовать для формирования такового впоследствии. Поэтому конъюнкция β поступает на вход блока «продолжение», где производится ее «достройка».

3. Для всех j $k_j < \bar{k}_j$. В этом случае конъюнкция β не может быть использована для формирования достаточно признака какого-либо класса с оценкой $k_j \geq \bar{k}_j$; она поступает на вход блока «провал», который перестраивает эту конъюнкцию, заменяя входящие в нее сомножители.

Блок «проверка» выполняет следующие две функции:

1. Если для класса A_i уже отобрано m признаков, то блок следит за тем, чтобы новая оценка \bar{k}_i была на еди-

ницу больше минимальной оценки признаков этого класса, записанных в памяти.

2. Решает вопрос о занесении или незанесении нового достаточного признака в память. Для уменьшения дисперсии числа отобранных признаков при занесении признака в память требуется, чтобы для него (как и для любого другого признака, уже записанного в память) существовали как примеры, обладающие первым и не обладающие вторым признаком, так и, наоборот, примеры, обладающие вторым и не обладающие первым признаком. При этом если окажется, что в памяти хранятся признаки, для которых множество примеров, ими обладающих, строго включает в себя множество примеров, обладающих новым признаком, то такие признаки стираются. Новый признак записывается на свободное место (если оно есть) или на место признака, имеющего минимальную оценку k_i , так как в этом случае оценка нового признака, по крайней мере, на единицу больше этой минимальной оценки.

Если когда-то был отобран такой признак, что множества примеров, обладающих этим и вновь отобранным признаком, совпадают, то в памяти хранится тот из них, который зависит от меньшего числа переменных, так как при этом есть надежда, что им обладает большее число объектов.

После описанной процедуры конъюнкция β поступает на вход блока «провал» для перестройки.

Блок «продолжение» проверяет, не достигла ли длина l рассматриваемой конъюнкции максимально допустимой длины \bar{l} . Если этого не произошло, то при условии, что последняя переменная конъюнкции β не является последней переменной кода объектов, т. е. при $i_r < n$ строится конъюнкция $\beta' = \beta X_{i_r+1}$, которую и будет рассматривать блок «анализ».

Если же длина конъюнкции β равна \bar{l} или индекс i_r последней переменной конъюнкции равен n , то конъюнкция передается для рассмотрения блоку «провал».

Блок «провал» осуществляет замену последнего множителя $x_{i_r}^{\sigma_r}$ конъюнкции β в том случае, когда эта конъюнкция не может быть больше использована при построении достаточных признаков с оценкой, не меньшей \bar{k}_i .

При этом возможны следующие случаи:

1. Конъюнкция β имеет вид $x_{i_1}^{\sigma_1} \wedge x_{i_2}^{\sigma_2} \wedge \dots \wedge x_{i_r}$, т. е. последняя переменная x_{i_r} входит в эту конъюнкцию без отрицания. В этом случае над последней переменной ставится знак отрицания и блок «анализ» рассматривает конъюнкцию:

$$\beta' = x_{i_1}^{\sigma_1} \wedge x_{i_2}^{\sigma_2} \wedge \dots \wedge \bar{x}_{i_r}.$$

2. Последняя переменная x_{i_r} конъюнкции β уже стоит под знаком отрицания, но индекс $i_r < n$, т. е. эта переменная не является последней переменной кода объектов. В этом случае сомножитель \bar{x}_{i_r} в конъюнкции β заменяется переменной, индекс которой на единицу больше, чем i_r , т. е. строится конъюнкция

$$\beta = x_{i_1}^{\sigma_1} \wedge x_{i_2}^{\sigma_2} \wedge \dots \wedge x_{i_r+1},$$

которая рассматривается блоком «анализ».

3. Последняя переменная x_{i_r} конъюнкции β стоит под знаком отрицания и его индекс $i_r = n$. В этом случае делается «шаг назад», т. е. последний сомножитель выбирается из конъюнкции β и полученная таким образом конъюнкция

$$\beta' = x_{i_1}^{\sigma_1} \wedge x_{i_2}^{\sigma_2} \wedge \dots \wedge x_{i_r-1}^{\sigma_{r-1}}$$

вновь рассматривается блоком «провал».

Обучение заканчивается после того, как будет рассмотрена конъюнкция \bar{x}_n .

Для дальнейшего уменьшения дисперсии числа отобранных признаков, характеризующих каждый из примеров, производится повторный цикл обучения (доучивание), т. е. дополнительный отбор признаков, который ведется следующим образом.

В каждом классе A_i выделяются примеры, для которых число отобранных признаков меньше некоторого порога v_i . Обычно $v_i = c_i M$, где M — среднее число признаков, характеризующих примеры, а c_i — некоторые константы, которые выбираются исходя из опыта. В этом случае после обучения алгоритм вычисляет значение M и автоматически устанавливает пороги v_i , используя кон-

станты c_i , которые должны быть заданы перед доучиванием.

После того как указанным образом выделено несколько примеров каждого класса, производится дополнительный отбор некоторого числа признаков, который отличается от отбора, проводимого при обучении, только тем, что оценка k_i отбираемого признака вычисляется лишь на множестве выделенных примеров. Таким образом, каждый из вновь отбираемых признаков должен охарактеризовать какие-то выделенные примеры, т. е. общее число признаков, которыми будут обладать эти примеры, увеличивается. Обычно такая процедура приводит к уменьшению дисперсии числа отобранных признаков, характеризующих примеры.

При необходимости доучивание может повторяться. Число последовательных доучиваний, константы c_i и количество признаков каждого класса, которые должны быть отобраны после каждого цикла доучивания, могут меняться в процессе работы алгоритма.

Следует отметить, что выбор начальных значений минимальных оценок \bar{k}_i^0 не влияет на то, какие именно признаки будут отобраны в процессе работы алгоритма (если, конечно, эти значения не настолько велики, что при этом вообще не может быть отобрано заданное количество признаков). Эти значения определяют лишь время перебора конъюнкций, так как при малых значениях оценок \bar{k}_i^0 большее число конъюнкций поступает на вход блока «продолжение». Поскольку выбор оптимальных (по времени) начальных значений \bar{k}_i^0 для каждого цикла обучения существенно зависит от вида задачи, то обычно и в начале обучения, и в начале доучивания эти значения полагаются равными единице. При этом несколько увеличивается время перебора конъюнкций.

Описанный алгоритм реализован в виде программы для вычислительной машины М-220. Программа может работать с двумя классами объектов A_1 и A_2 . Число примеров каждого класса, используемых при обучении, может быть не более 180. Для формирования оценок признаков используется не более 45 выделенных примеров каждого класса (остальные примеры необходимы лишь для проверки условий достаточности отбираемых признаков). Отобранный признак является признаком данного класса только в том случае, если им обладает не

менее \bar{k}_i примеров из 45 выделенных примеров этого класса и ни один из примеров противоположного класса.

Перед началом работы программы в машину вводятся перфокарты с кодами примеров. Каждый код занимает одну 45-разрядную ячейку, так что значению каждого параметра соответствует один разряд.

В начале работы программы весь массив примеров каждого класса разбивается на группы по 45 примеров в каждой. Образованные при таком разбиении бинарные матрицы 45×45 поворачиваются так, что значения каждой переменной по всем 45 примерам данной группы записываются в одну ячейку. Теперь значения любой конъюнкции этих переменных могут быть получены одновременно для всех 45 примеров групп, если использовать команды поразрядных логических операций. Значение любой конъюнкции переменных x_1, x_2, \dots, x_n по этим 45 примерам также, очевидно, будет записано в одной ячейке. В начале каждого цикла обучения начальные значения минимальных оценок \bar{k}_i^0 отбираемых признаков записываются в соответствующие рабочие ячейки памяти.

В дальнейшем величины оценки признаков будут сравниваться с содержимым этих рабочих ячеек.

В процессе перебора конъюнкций вместе с рассматриваемой конъюнкцией $x_{i_1}^{\sigma_1} \wedge x_{i_2}^{\sigma_2} \wedge \dots \wedge x_{i_r}^{\sigma_r}$ в r последовательных ячейках памяти хранится вся цепочка конъюнкций, приводящих к данной, а именно:

$$\begin{aligned} & x_{i_1}^{\sigma_1}; \\ & x_{i_1}^{\sigma_1} \wedge x_{i_2}^{\sigma_2}; \\ & \dots \dots \dots \\ & x_{i_1}^{\sigma_1} \wedge x_{i_2}^{\sigma_2} \wedge \dots \wedge x_{i_r}^{\sigma_r}. \end{aligned}$$

Хранение в памяти ЭВМ такой цепочки позволяет избежать повторного построения конъюнкций, когда блоку «провал» приходится делать «шаг назад». Информация об отбираемых признаках хранится в четырех ячейках памяти: в первой ячейке хранится информация о том, функцией каких переменных является данный признак. Каждой переменной соответствует определенный разряд ячейки (единицы ставятся в тех разрядах, которые соответствуют переменным, входящим в данный

признак). Во второй ячейке единицы записываются в разряды, соответствующие переменным, входящим в этот признак без отрицаний. В третьей ячейке единицами в разрядах указаны номера тех примеров (из первых 45), на которых конъюнкция принимает единичное значение. Содержание этой ячейки определяется непосредственно при вычислении значения конъюнкции. Наконец, в четвертой ячейке хранится оценка признака, т. е. число примеров, номера которых указаны в третьей ячейке. Если в системе команд машины имеется команда счета единиц в ячейке, то можно ограничиться запоминанием содержимого первых трех ячеек, так как содержимое четвертой ячейки может быть получено в результате применения этой команды к содержимому третьей ячейки; время перебора возрастает незначительно.

СПИСОК ЛИТЕРАТУРЫ

1. Вайнцвайг М. Н. Об одном алгоритме распознавания двоичных знаков. «Проблемы передачи информации», т. 2, 1966, вып. 3.

В. Л. Браиловский, А. Л. Лунц

Программа обучения распознаванию, основанная на отборе статистически полезных признаков

Описываемые здесь алгоритм и программа, его реализующая предназначены для решения научно-технических и практических задач распознавания образов.

Накопленный опыт показал, что статистические методы, применяемые в программе, повышают помехоустойчивость алгоритма, а также позволяют получить приемлемое качество решения в сложных задачах.

Предлагаемое описание программы предназначено прежде всего для решения конкретных задач. В соответствии с этим основное внимание уделено описанию алгоритма и особенностям программирования, существенным для эффективной программной реализации, а также некоторым практическим рекомендациям. Вопросы обоснования алгоритма рассмотрены в [1—4].

1. ОПИСАНИЕ АЛГОРИТМА

Алгоритм оперирует векторами, каждый из которых характеризуется совокупностью параметров $\xi_1, \xi_2, \dots, \xi_n$, и, следовательно, представляется точкой в n -мерном пространстве R_n . Каждый параметр ξ_i может принимать лишь конечное число значений $q_i: 0, 1, \dots, (q_i-1)$; следовательно, все пространство состоит из конечного числа точек.

Первый режим работы алгоритма «обучение» состоит из следующих основных процедур:

1. *Рассмотрение проекций обучающей выборки.* Отбираются наиболее информативные комбинации параметров (подпространства). Обучающая выборка векторов Q проектируется во всевозможные координатные подпространства, размерность которых не выше некоторого m (обычно $m=2 \div 4$). Так, пусть, например, рассматривается двумерное ($m=2$) подпространство, построенное на параметрах ξ_i и ξ_j , имеющих соответственно q_i и q_j возможных значений. Число сочетаний значений, которое может принимать пара параметров (ξ_i, ξ_j) , равно $q_i \times q_j$; столько же точек содержится в рассматриваемом подпространстве.

Для каждого вектора обучающей выборки Q , в соответствии со значениями его параметров ξ_i и ξ_j , выясняется, в какую точку этого подпространства он попадает при проектировании. Для любой точки подпространства s после выполнения этой операции над векторами выборки Q известно, сколько векторов класса A (из общего числа N_A) и класса B (из общего числа N_B) в нее спроектировано.

Пусть это будут соответственно k_s и l_s . Значения вероятностей для точки s можно оценить с помощью метода Байеса:

$$\hat{P}(s/A) = (k_s + \Delta) / (N_A + 2\Delta), \quad (1.1)$$

$$\hat{P}(s/B) = (l_s + \Delta) / (N_B + 2\Delta), \quad (1.2)$$

где $\hat{P}(s/A)[\hat{P}(s/B)]$ — оценка вероятности попадания вектора в точку s при условии принадлежности его

к классу $A[B]$; Δ — параметр алгоритма*), чаще всего равный единице.

Для каждой точки s вычисляется оценка вероятности принадлежности вектора классу $A[B]$ при условии, что он проектируется в эту точку:

$$\hat{P}(x \in A) = \frac{\mu(A) \hat{P}(s/A)}{\mu(A) \hat{P}(s/A) + \mu(B) \hat{P}(s/B)}, \quad (1.3)$$

$$\hat{P}(x \in B) = 1 - \hat{P}(x \in A). \quad (1.4)$$

Здесь $\mu(A)[\mu(B)]$ — априорная вероятность принадлежности вектора классу $A[B]$. Эти величины также являются параметрами алгоритма.

Для решения задачи назначается система цен $\{\lambda\}$, определяющих стоимость правильных и неправильных ответов:

λ_{AA} — выигрыш, получаемый, если вектор класса A правильно опознан,

λ_{AB} — выигрыш, получаемый, если вектор класса A опознан как вектор класса B ,

λ_{BA} — выигрыш, получаемый, если вектор класса B опознан как вектор класса A ,

λ_{BB} — выигрыш, получаемый, если правильно опознан вектор класса B .

Совокупность параметров $\{\lambda\}$ определяется исходя из требований, связанных с конкретной решаемой задачей.

Каждому вектору из обучающей выборки Q в рассматриваемом подпространстве соответствуют величины $\hat{P}(x \in A)$ и $\hat{P}(x \in B)$, полученные при проектировании. Если этот вектор принадлежит классу A , то оценка качества его опознания в подпространстве

$$z_A(x) = \lambda_{AA} \hat{P}(x \in A) + \lambda_{AB} \hat{P}(x \in B). \quad (1.5)$$

Для вектора, принадлежащего классу B , соответственно

$$z_B(x) = \lambda_{BA} \hat{P}(x \in A) + \lambda_{BB} \hat{P}(x \in B). \quad (1.6)$$

*) За счет изменения параметра Δ можно варьировать «степень доверия» к наблюдаемому в точке соотношению числа векторов класса A и класса B (k_s и l_s) в зависимости от «зашумленности» обрабатываемой информации. Заметим, что при $\Delta=0$ формулы (1.1) и (1.2) дают оценку вероятности с помощью соответствующих частот; $\Delta=1$ получается при равномерном априорном распределении вероятностей $P(s/A)$ и $P(s/B)$.

Ценность подпространства вычисляется как среднее по векторам:

$$z = \frac{\mu(A)}{N_A} \sum_{x \in Q_A} z_A(x) + \frac{\mu(B)}{N_B} \sum_{x \in Q_B} z_B(x). \quad (1.7)$$

Здесь $Q_A [Q_B]$ — обучающая выборка векторов класса $A [B]$; $Q = Q_A \cup Q_B$.

Таким образом, оказывается возможным каждому подпространству из рассматриваемой совокупности поставить в соответствие величину его «ценности», вычисляемую по формуле (1.7). В результате перебора отбирается L подпространств, обладающих наибольшей ценностью (обычно $L \approx 100-150$). L также является параметром алгоритма.

2. *Вычисление ценности совокупности двух подпространств.* Предварительно для отобранных подпространств вычисляется отношение правдоподобия. Для каждого вектора x по подпространству T оценка отношения вероятностей принадлежности классу A и B , как следует из выражений (1.1) — (1.4), равна:

$$\hat{L}_T = \frac{\mu(A)}{\mu(B)} \hat{L}_{T0} = \frac{\mu(A)}{\mu(B)} \frac{k_s + \Delta}{l_s + \Delta} \frac{N_B + 2\Delta}{N_A + 2\Delta}. \quad (1.8)$$

Здесь k_s и l_s — числа векторов класса A и B , попавших в ту же точку s , куда и вектор x . Практически в программе вычисляется величина $\log \hat{L}_{T0}$, которая рассчитывается по формуле, использующей пороговую отсечку θ , а именно:

$$\log \hat{L}_{T0} = \begin{cases} 0, & \text{если } \left| \log \left(\frac{k_s + \Delta}{l_s + \Delta} \frac{N_B + 2\Delta}{N_A + 2\Delta} \right) \right| < \theta, \\ \log \left(\frac{k_s + \Delta}{l_s + \Delta} \frac{N_B + 2\Delta}{N_A + 2\Delta} \right) & \text{в противном случае} \end{cases}$$

$$\hat{L}_T = [\mu(A)/\mu(B)] \hat{L}_{T0}. \quad (1.8')$$

Здесь θ — параметр программы (часто выбирается равным единице). Этот эмпирический прием приводит к тому, что классификация вектора производится в дальнейшем лишь по совокупности тех подпространств, в каждом из которых вектор классифицируется достаточно определенно. Тем самым ослабляется влияние зашумлен-

ности» из-за ограниченного объема выборки, погрешности в измерении параметров, ошибки в классификации векторов обучающей выборки и т. д.

Векторы, у которых:

$0 < \hat{L}_T \leq 1/3$ образуют группу Γ_1 ;

$1/3 < L_T < 3$ образуют группу Γ_2 ;

$L_T \geq 3$ образуют группу Γ_3 .

Пусть в подпространстве T_1 вектор x имеет оценку отношения вероятностей \hat{L}_{T_1} и попадает в группу Γ_i^1 , а в подпространстве T_2 — оценку \hat{L}_{T_2} и попадает в группу Γ_j^2 .

Тогда, как показано в работах [2, 3], отношение вероятности принадлежности к классу A к вероятности принадлежности к классу B для вектора определяется по формуле

$$L_{T_1 T_2} = L_{T_1} L_{T_2} \frac{P(\Gamma_j^2 / \Gamma_i^1 \cap A)}{P(\Gamma_j^2 / \Gamma_i^1 \cap B)} \frac{P(B / \Gamma_j^2)}{P(A / \Gamma_j^2)}, \quad (1.9)$$

где $P(\Gamma_j^2 / \Gamma_i^1 \cap A)$ — вероятность попаданий вектора в группу Γ_j^2 по подпространству T_2 при условии, что по подпространству T_1 этот вектор попадает в группу Γ_i^1 и относится к классу A ; $P(B / \Gamma_j^2)$ — вероятность принадлежности вектора к классу B при условии попадания его в группу Γ_j^2 по подпространству T_2 .

Аналогично определяются величины $P(\Gamma_j^2 / \Gamma_i^1 \cap B)$ и $P(A / \Gamma_j^2)$.

Для получения оценки $\hat{L}_{T_1 T_2}$ в формуле (1.9) вместо величин L_{T_1} и L_{T_2} подставляются их оценки, вычисленные по формуле (1.8'). Оценки остальных величин, входящих в выражение (1.9), вычисляются следующим образом:

$$\hat{P}(A / \Gamma_j^2) = \frac{\mu(A)}{\mu(A) N_{A_j} + \mu(B) N_{B_j}} \sum_{x \in \Gamma_j^2 \cap Q_A} \hat{P}(x \in A) +$$

$$+\frac{\mu(B)}{\mu(A)N_{A_j}+\mu(B)N_{B_j}}\sum_{x\in\Gamma_i^2\cap Q_B}\hat{P}(x\in A); \quad (1.10)$$

$$\begin{aligned} \hat{P}(B/\Gamma_j^2) &= \frac{\mu(A)}{\mu(A)N_{A_j}+\mu(B)N_{B_j}}\sum_{x\in\Gamma_j^2\cap Q_A}\hat{P}(x\in B) + \\ &+ \frac{\mu(B)}{\mu(A)N_{A_j}+\mu(B)N_{B_j}}\sum_{x\in\Gamma_j^2\cap Q_B}\hat{P}(x\in B). \end{aligned} \quad (1.11)$$

Здесь N_{A_j} и N_{B_j} — количество векторов соответственно классов A и B , попавших в группу Γ_j^2 ; оценки вероятностей $\hat{P}(x\in A)$ и $\hat{P}(x\in B)$ вычисляются по формулам (1.3) и (1.4) для той точки, в которую попал вектор в подпространстве T_2 .

Оценки величин $P(\Gamma_j^2/\Gamma_i^1 \cap A)$ и $P(\Gamma_j^2/\Gamma_i^1 \cap B)$ вычисляются по следующим формулам:

$$\hat{P}^*(\Gamma_j^2/\Gamma_i^1 \cap A) = \frac{D\hat{P}(\Gamma_j^2/A) + k_{ij}}{D + N_{A_i}}, \quad (1.12)$$

$$\hat{P}^*(\Gamma_j^2/\Gamma_i^1 \cap B) = \frac{D\hat{P}(\Gamma_j^2/B) + l_{ij}}{D + N_{B_i}}, \quad (1.13)$$

$$\hat{P}(\Gamma_j^2/\Gamma_i^1 \cap A) = \frac{\hat{P}^*(\Gamma_j^2/\Gamma_i^1 \cap A)}{\sum_j \hat{P}^*(\Gamma_j^2/\Gamma_i^1 \cap A)}; \quad (1.14)$$

$$\hat{P}(\Gamma_j^2/\Gamma_i^1 \cap B) = \frac{\hat{P}^*(\Gamma_j^2/\Gamma_i^1 \cap B)}{\sum_j \hat{P}^*(\Gamma_j^2/\Gamma_i^1 \cap B)}. \quad (1.15)$$

$$\hat{P}(\Gamma_j^2/A) = \frac{\hat{P}(A/\Gamma_j^2) \hat{P}(\Gamma_j^2)}{\mu(A)}; \quad (1.16)$$

$$\hat{P}(\Gamma_j^2/B) = \frac{\hat{P}(B/\Gamma_j^2) \cdot \hat{P}(\Gamma_j^2)}{\mu(B)}; \quad (1.17)$$

$$\hat{P}(\Gamma_j^2) = \frac{N_{A_j}}{N_A} \mu(A) + \frac{N_{B_j}}{N_B} \mu(B). \quad (1.18)$$

Формулы (1.12), (1.13) являются формулами Байеса с априорными значениями оценок вероятностей $\hat{P}(\Gamma_j^2/A)$ и $\hat{P}(\Gamma_j^2/B)$, вычисляемыми по формулам (1.16) — (1.18).

Здесь D — параметр алгоритма (обычно $D=4$); $k_{ij}(l_{ij}$ — количество векторов класса $A[B]$, принадлежащих одновременно группам Γ_i^1 и Γ_j^2 ; $N_{A_i}[N_{B_i}]$ — количество векторов класса $A[B]$, принадлежащих группе Γ_i^1 , $N_{A_j}[N_{B_j}]$ — то же для группы Γ_j^2 .

Равенства (1.14), (1.15) — условия нормирования.

Учитывая выражения (1.10) — (1.18), можно по формуле (1.9) получить для каждого вектора оценку величины отношения вероятностей принадлежности вектора к классу A и B по совокупности двух подпространств. Далее с учетом условия нормирования определяются значения оценок вероятностей $\hat{P}(x \in A)$ и $\hat{P}(x \in B)$ для вектора x по совокупности подпространств T_1 и T_2 .

Учитывая, все вышесказанное, можно получить для каждого вектора оценки вероятности $\hat{P}(x \in A)$ и $\hat{P}(x \in B)$ по любому количеству подпространств, последовательно увеличивая их совокупность. Так, при вычислении оценок по совокупности трех подпространств T_1 , T_2 и T_3 в формулу (1.9) вместо L_{T_1} следует подставить величину $L_{T_1 T_2}$, способ вычисления которой описан выше, а вместо величины L_{T_2} — величину L_{T_3} . Группы Γ_i^1 образуются в зависимости от значений $L_{T_1 T_2}$, а группы Γ_j^2 — в зависимости от значений. Заметим также, что, получив для каждого вектора x оценки $\hat{P}(x \in A)$ и $\hat{P}(x \in B)$ по некоторой совокупности подпространств, с помощью формул (1.5) — (1.7) можно получить оценку ценности этой совокупности подпространств.

3. *Отыскание совокупности подпространств, содержащей наиболее полную (или близкую к таковой) информацию о правиле классификации.* Выбирается подпространство, обладающее наибольшей ценностью (см. п. 1), и к нему поочередно присоединяется каждое подпрост-

ранство из списка наиболее ценных *). Наиболее ценная пара подпространств фиксируется; затем к ней наилучшим образом присоединяется третье подпространство и т. д.

Процесс заканчивается тогда, когда ценность вновь образовавшейся группы не превосходит (или превосходит незначительно) ценность группы подпространств, образованной на предыдущем шаге работы алгоритма. Отобранную таким образом совокупность подпространств назовем *оптимальным набором подпространств*.

4. *Второй режим работы алгоритма — классификация* (распознавание) — строится следующим образом. Вектор x , поступивший для классификации, проектируется в каждое из отобранных в оптимальный набор подпространств; при этом выясняется, какую оценку \hat{L} он получает в каждом таком подпространстве. Затем вычисляется оценка \hat{L} на совокупности отобранных в оптимальный набор подпространств, что позволяет получить оценки вероятностей $\hat{P}(x \in A)$ и $\hat{P}(x \in B)$ по этому набору. С помощью выражения (1.5) вычисляется средний выигрыш, который может быть получен при отнесении этого вектора к классу A , а с помощью выражения (1.6) — средний выигрыш, получаемый при отнесении его к классу B .

Ответ о принадлежности вектора к тому или иному классу дается по следующему правилу:

а) если $z_A(x) - z_B(x) \geq D_A$, то $x \in A$,

б) если $z_B(x) - z_A(x) > D_B$, то $x \in B$,

в) если $-D_B \leq [z_A(x) - z_B(x)] < D_A$,

то ответ неопределенный.

Здесь D_A, D_B — неотрицательные константы, характеризующие пороги классификации, которые для каждой задачи выбираются экспериментально.

2. ОПИСАНИЕ ПРОГРАММЫ

Алгоритм, рассмотренный в § 1, запрограммирован для машин типа М-220. Программа предусматривает

*) Под присоединением одного подпространства к другому или к совокупности подпространств понимается вычисление для каждого вектора оценок вероятностей $P(x \in A)$ и $P(x \in B)$ по этим подпространствам и вычисление ценности этих подпространств по формулам (1.5) — (1.7).

обработку до 400—500 векторов, каждый из которых описывается несколькими десятками параметров. Эта программа состоит из 1 500 команд, объединенных в 35 подпрограмм. Все подпрограммы организованы в четыре блока, которые объединяются управляющей программой.

Кодирование векторов и задание информации о них. Все векторы как из обучающей, так и из экзаменационной выборки имеют стандартную форму. Вектор характеризуется параметрами, которые могут быть выражены одно-, двух- и трехразрядными числами (в конкретной задаче разрядность всех таких чисел одинакова).

Вектор может занимать одну или две ячейки памяти. Поэтому предельное число параметров в описании вектора есть 90, 44 или 30 при их разрядности соответственно 1, 2 или 3. Максимальное число значений, принимаемых одним параметром, соответственно равно 2, 4 или 8.

Во входной информации указывается число значений, фактически принимаемых каждым параметром, а также разрешен или запрещен этот параметр к использованию.

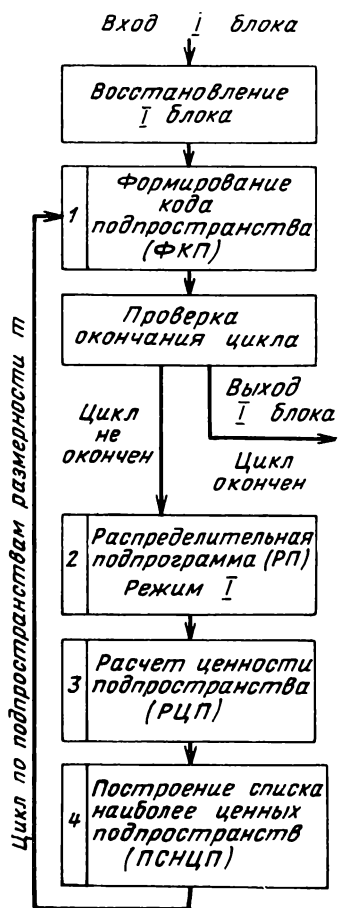


Рис. 1.

Описание I блока программы

I блок программы выполняет процедуру, описанную в п. 1 § 1, а именно осуществляет рассмотрение проекций обучающей выборки Q во все подпространства размерно-

сти не выше m и оценку информативности каждого из них и, кроме того, производит отбор совокупности из L наиболее информационных подпространств. Блок-схема этого блока приведена на рис. 1.

Блок состоит из следующих подпрограмм.

1. Подпрограмма формирования кода подпространства (ФКП) образует совокупность номеров параметров, являющихся координатами рассматриваемого подпространства. Эта совокупность называется кодом подпространства. После формирования последнего сочетания номеров эта подпрограмма вырабатывает сигнал окончания работы I блока. В формировании кода подпространства принимают участие лишь разрешенные параметры (в соответствии с входной информацией).

2. Распределительная подпрограмма (РП) вычисляет распределение векторов по точкам подпространства, код которого сформирован предыдущей подпрограммой. Это режим I работы РП (имеется также режим II, о котором будет сказано ниже). РП в режиме I вычисляет данные на поле распределения, которое устроено таким образом, что каждой точке подпространства соответствует определенная ячейка памяти. Адрес ячейки поля распределения (Ad) связан с координатами точки подпространства соотношением

$$Ad(\xi_{p_1}, \xi_{p_2}, \dots, \xi_{p_m}) = 0006 + \xi_{p_m} \prod_{i=1}^{m-1} q_{p_i} + \\ + \xi_{p_{m-1}} \prod_{i=1}^{m-2} q_{p_i} + \dots + \xi_{p_1}. \quad (2.1)$$

Здесь ξ_{p_i} — значение p_i -го параметра; $0 \leq \xi_{p_i} \leq q_{p_i} - 1$.

Все вычисления в выражении (2.1) проводятся в восьмеричной системе счисления. Перед началом работы РП в режиме I поле распределения очищается записью во все ячейки нулей. Подпрограмма выбирает последовательно векторы класса A , отбирает из описания каждого вектора параметры $\xi_{p_1}, \dots, \xi_{p_m}$ в соответствии с кодом подпространства, а затем по формуле (2.1) вычисляет адрес ячейки на поле распределения. К содержимому левого адреса этой ячейки добавляется единица. После рассмотрения всех векторов обучающей выборки класса A программа переходит к аналогичной

процедуре для класса B с той лишь разницей, что единица добавляется к содержимому среднего адреса ячейки, соответствующей точке подпространства, в которую попал этот вектор.

Нетрудно показать, что скорость работы блока I программы почти целиком определяется скоростью работы РП, поэтому ее следует программировать особенно тщательно.

Выходной информацией подпрограммы РП в режиме I является поле распределения, где в левом адресе каждой ячейки записано число векторов класса A , а в среднем — число векторов класса B , попавших при проектировании в соответствующую точку подпространства [см. формулу (2.1)].

3. Подпрограмма расчета ценности подпространства (РЦП) вычисляет ценность подпространства в соответствии с выражением (1.7). Входной информацией для этой подпрограммы является поле распределения, выработанное РП в режиме I.

Подпрограмма для каждой точки — ячейки поля распределения проводит вычисления по формулам (1.1) — (1.6), на основании которых рассчитывается ценность подпространства по формуле (1.7).

4. Подпрограмма построения списка наиболее ценных подпространств (ПСНЦП) отбирает «ценности» и коды L наиболее информативных (ценных) подпространств, рассмотренных к данному моменту времени. Эта информация хранится в памяти ЭВМ в виде двух списков: списка ценностей подпространств, который всегда упорядочен в порядке убывания цен, и списка соответствующих им кодов подпространств.

Перед началом работы I блока происходит восстановление и настройка блока по параметрам решаемой задачи. Очередное прохождение цикла по подпространствам размерности m выглядит следующим образом. Подпрограмма ФКП вырабатывает новый код подпространства. Управляющая программа с помощью операции «Проверка окончания цикла» выясняет, выдан ли сигнал окончания работы I блока. В случае положительного ответа работа этого блока заканчивается, в противном случае следует обращение к подпрограмме РП. Последняя на поле распределения вычисляет распределение векторов по точкам рассматриваемого подпространства. Далее по этому распределению подпрограмма

РЦП вычисляет ценность подпространства, а подпрограмма ПСНЦП определяет, подлежат ли код и ценность этого подпространства «текущему» запоминанию. После этого управление вновь передается на подпрограмму ФКП.

В результате работы I блока программы в памяти машины образуется список кодов L наиболее ценных подпространств и список их ценностей. Реально $L \approx 100 \div 150$.

Описание II блока программы

Во II блоке описание векторов выборки с помощью исходной системы параметров заменяется их описанием с помощью отношений правдоподобия (точнее, их оценок), вычисленных в соответствии с формулой (1.8') по группе отобранных подпространств.

В режиме обучения обучающая выборка перекодируется, и основной цикл II блока выполняется L раз, в соответствии со списком подпространств, отобранных I блоком. Во II блок включены следующие подпрограммы:

1. Распределительная подпрограмма РП в режиме I строит распределение векторов обучающей выборки по точкам очередного подпространства, указанного в списке наиболее ценных подпространств.

2. Подпрограмма определения ценности для каждой точки подпространства (ЦТ) по формуле (1.8') вычисляет величину $\log_2 \hat{L}_{To}$, а затем $\log_2 \hat{L}_T$, которая далее кодируется 5-разрядным двоичным кодом, обозначаемым \bar{L} . Выходной информацией подпрограммы является поле распределения с вписанной в каждую ячейку величиной \bar{L} . Эта же подпрограмма вычисляет по формуле (1.10) вспомогательные величины $\hat{P}(A/\Gamma_j^2)$ (см. п. 2, § 1), $j=1, 2, 3$, используемые лишь в III блоке. Каждая такая величина кодируется 5-разрядным двоичным кодом.

3. Далее работает распределительная подпрограмма (РП) в режиме II (перекодирования). Эта подпрограмма использует информацию, полученную на поле распределения подпрограммой ЦТ. Для каждого вектора, подлежащего перекодированию, определяется адрес ячейки, соответствующей точке подпространства, в ко-

торую он проектируется [см. формулу (2.1)]. Затем величина \bar{L} , вычисленная для этой ячейки, переписывается в ячейку, где формируется новый код вектора.

Так как величина \bar{L} занимает пять двоичных разрядов, то в каждой ячейке памяти помещается девять величин такого типа, описывающих значения \bar{L} , полученные данным вектором по девяти последовательным подпространствам списка.

В заключение заметим, что векторы, подлежащие перекодированию, могут как совпадать, так и не совпадать с обучающей выборкой.

В случае, когда II блок работает со списком подпространств, отобранных I блоком, перекодированию подлежит именно обучающая выборка.

Общая блок-схема работы II блока программы представлена на рис. 2.

В результате работы I блока получен список кодов наиболее ценных пространств (далее, при описании IV блока в качестве такого списка будет использован оптимальный набор подпространств, отобранный III блоком). Управляющая программа выбирает очередное подпространство списка и по нему с помощью РП в режиме I строится поле распределения векторов обучающей выборки; далее для каждой ячейки этого поля с помощью подпрограммы ЦТ вычисляется величина \bar{L} , которая записывается в эту же ячейку.

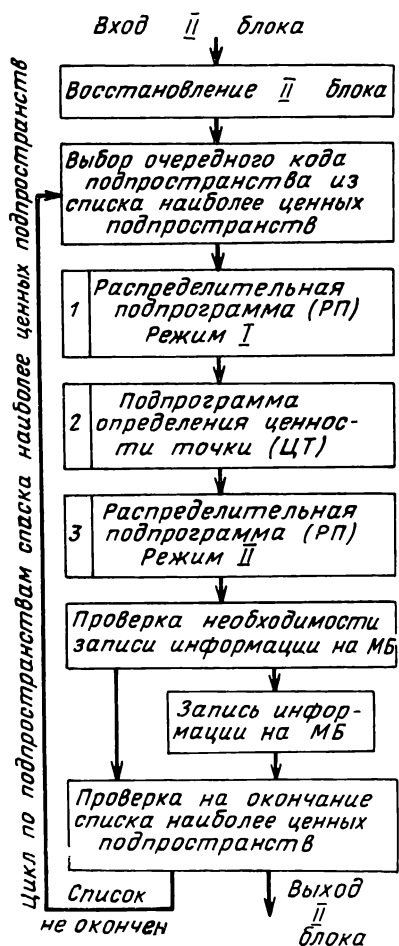


Рис. 2.

Подпрограмма РП в режиме II для каждого вектора, подлежащего перекодированию (в режиме обучения — это обучающая выборка, в режиме экзамена — экзаменационная), по формуле (2.1) вычисляет адрес ячейки поля распределения, а величину \tilde{L} , записанную в этой ячейке в виде 5-разрядного двоичного кода, переписывает в ячейку, где формируется новый код этого вектора. Эти ячейки образуют поля A_{Π} , B_{Π} . После того как рассмотрено девять подпространств, все 45 разрядов каждой ячейки полей A_{Π} , B_{Π} заполняются, поля A_{Π} , B_{Π} переписываются на МБ, образуя очередной массив, а затем во все ячейки этих полей записываются нули.

Управляющая программа II блока после окончания работы РП в режиме II проверяет необходимость переписи информации с полей A_{Π} , B_{Π} на МБ и в случае необходимости выполняет эту процедуру. Далее управляющая программа проверяет, окончился ли список наиболее ценных подпространств. В случае отрицательного ответа процесс рассмотрения подпространств возобновляется. После рассмотрения последнего подпространства информация, записанная на поля A_{Π} , B_{Π} , переписывается на МБ, и работа II блока заканчивается. По окончании работы блока на МБ остаются перекодированные векторы, расположенные по массивам.

Описание III блока программы

Наиболее сложным блоком программы является III блок; его блок-схема приведена на рис. 3. Этот блок предназначен для образования оптимального набора подпространств методами, изложенными в § 1, пп. 2, 3; он состоит из следующих подпрограмм и процедур.

1. Подпрограмма установления начального состояния (УНС) в качестве первого подпространства оптимального набора выбирает первое (наиболее ценное) подпространство из списка L подпространств, образованного I блоком программы; затем эта подпрограмма определяет, в какую группу Γ^i по первому подпространству попал каждый рассматриваемый вектор.

В результате работы подпрограммы УНС для каждого вектора (при обучении — это векторы обучающей выборки, а при распознавании — экзаменационной) в ячейке полей A' , B' запоминается величина \tilde{L} и номер группы по первому подпространству.

2. Группа подпрограмм осуществляет процедуру объединения подпространства с совокупностью ранее отобранных в оптимальный набор (ОПС). При этом для каждого вектора вычисляется величина \bar{L} по новой совокупности подпространств с помощью метода, описанного

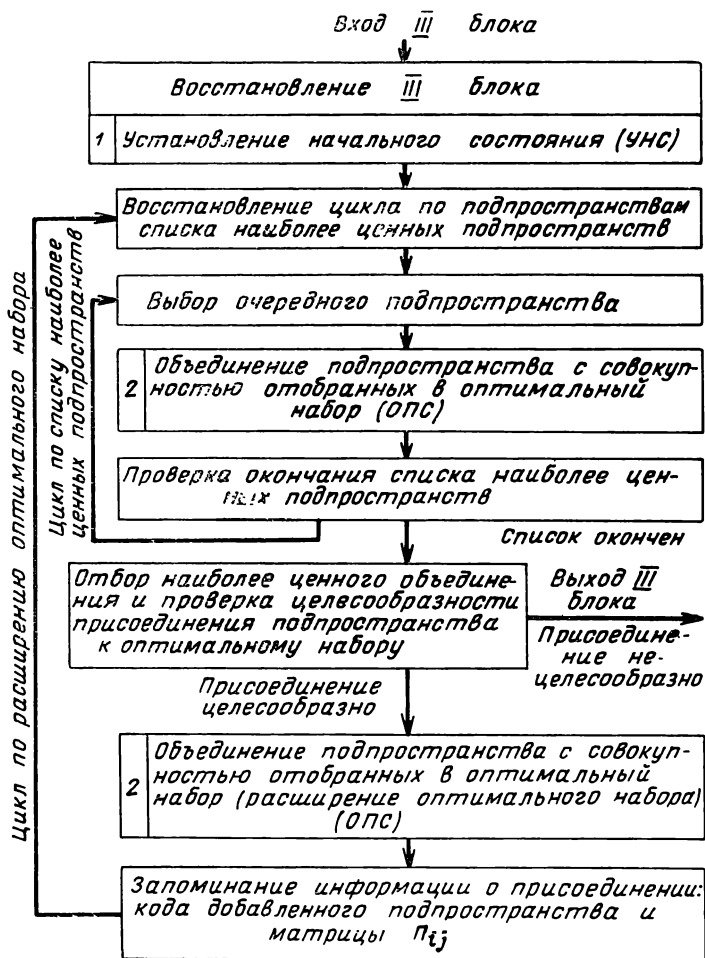


Рис. 3.

в § 1, п. 2. Эта группа подпрограмм использует следующую входную информацию. Для каждого вектора выборки (обучающей или экзаменационной) в ячейке полей

A', B' содержится величина \tilde{L} , вычисленная по уже отобранной совокупности подпространств и номеру группы Γ^1_j , в которую попал вектор в соответствии со значением \tilde{L} .

Рассматриваемая процедура выполняется с помощью следующих подпрограмм, приведенных на рис. 4.

Подпрограмма считывания массива с МБ на поля A_{Π}, B_{Π} считывает массив, содержащий для каждого вектора величины \tilde{L} , вычисленные по присоединяемому подпространству (см. описание II блока программы).

Подпрограмма вычисления номера группы для каждого вектора определяет, в какую группу Γ^2_j попала величина \tilde{L} , вычисленная по присоединяемому подпространству.

Подпрограмма «Цикл-1» выполняет следующие операции:

а) для каждого вектора вычисляется сумма величин \tilde{L} по уже отобранной совокупности подпространств и величине \tilde{L} , рассчитанной по присоединяемому подпространству;

б) одновременно вычисляется количество векторов классов A и B (в отдельности), попавших в i -ю группу Γ^1_i , по уже накопленному набору подпространств, и в группу Γ^2_j по присоединяемому подпространству для всевозможных комбинаций i, j (в количестве девяти).

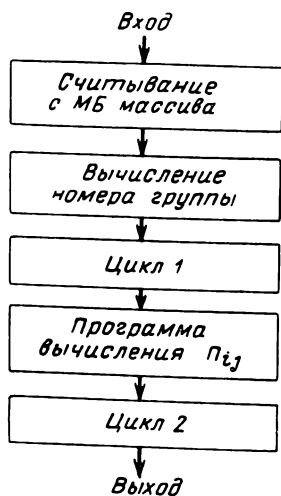


Рис. 4.

Подпрограмма вычисления Π_{ij}

Информации, полученной предыдущей подпрограммой, достаточно для того чтобы для каждого сочетания групп i, j вычислить поправку, которую следует добавить к сумме логарифмов $\lg_2 L_{T_1}$ и $\lg_2 L_{T_2}$ [см. выражение (1.9)] для получения $\lg_2 L_{T_1 T_2}$.

Эта поправка равна

$$\Pi_{ij} = \log \frac{\hat{P}(\Gamma_j^2/\Gamma_i^1 \cap A)}{\hat{P}(\Gamma_j^2/\Gamma_i^1 \cap B)} + \log_2 \frac{\hat{P}(B/\Gamma_j^2)}{\hat{P}(A/\Gamma_j^2)}.$$

Оценки величин $\hat{P}(A/\Gamma_j^2)$ и $\hat{P}(B/\Gamma_j^2)$, вычисляемые по формулам (1.10), (1.11) для каждого подпространства, были получены в блоке II. Величина поправки Π_{ij} кодируется 5-разрядным двоичным кодом.

Подпрограмма «Цикл-2» для каждого вектора к сумме величин \bar{L} , вычисленной подпрограммой «Цикл-1», добавляет поправку Π_{ij} . Далее вычисляются значения $\hat{P}(x \in A)$ и $\hat{P}(x \in B)$ по совокупности подпространств вместе с присоединяемым подпространством.

Применение выражений (1.5) — (1.7) позволяет определить ценность новой совокупности подпространства. На этом процедура ОПС заканчивается.

Работа блока III программы в режиме обучения организуется следующим образом.

Выбирается очередное подпространство из списка наиболее ценных подпространств, отобранных блоком I и с помощью процедуры ОПС оно объединяется с совокупностью, отобранной в оптимальный набор; далее вычисляется ценность новой совокупности подпространств. После завершения цикла по списку наиболее ценных подпространств отбирается подпространство, которое в сочетании с уже отобранной в оптимальный набор совокупностью имеет наибольшую ценность. Если эта ценность превышает ценность, полученную по ранее отобранной совокупности подпространств, более чем на заданную величину Δz , то это подпространство включается в оптимальный набор и образуется новая совокупность подпространств. Цикл по расширению оптимального набора блока III осуществляет присоединение нового подпространства к прежней совокупности и вычисляет для каждого вектора значения \bar{L} и номер группы i по новой совокупности подпространств. Эта информация записывается в соответствующие разряды ячеек полей A' , B' .

В том случае, если по окончании выполнения цикла по списку наиболее ценных подпространств выясняется, что ценность новой совокупности превосходит ценность старой совокупности подпространств менее чем на задан-

ную величину Δz , присоединение нового подпространства не происходит. Совокупность фиксируется; она и образует оптимальный набор подпространств; на этом работа блока III заканчивается. В ходе выполнения III блока программы на МБ запоминаются коды подпространств, образующих оптимальный набор, и совокупность матриц Π_{ij} , полученных при каждом новом пополнении совокупности подпространств.

Описание блока IV программы

Функции блока IV программы заключаются в том, чтобы для N_p векторов, поступивших для распознавания (экзамена), вычислить оценку вероятности принадлежности вектора к тому или иному классу и классифицировать каждый из этих векторов в соответствии с максимальным ожидаемым выигрышем [см. выражение (1.19)]. Блок-схема блока IV программы приведена на рис. 5. Этот блок использует обучающую выборку векторов, а также информацию, полученную блоком III программы. При этом осуществляются следующие процедуры.

1. С помощью блока II программы производится перекодирование векторов экзаменационной выборки; каждому вектору приписываются величины оценок отношения правдоподобия, вычисленные по (1.8') для каждого подпространства из оптимального набора, образованного блоком III программы *).

2. С помощью цикла III блока по расширению оптимального набора для каждого из распознаваемых векторов вычисляется величина \tilde{L} по совокупности подпро-

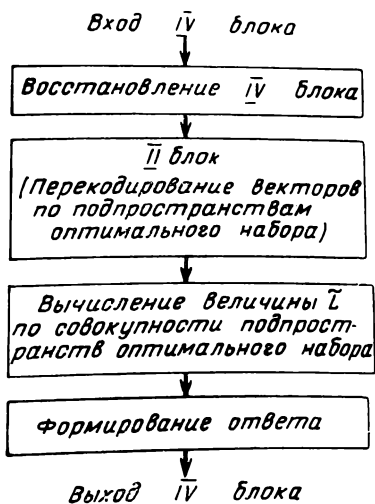


Рис. 5.

*) Величины, подставляемые в формулу (1.8'), вычисляются по обучающей выборке.

странств оптимального набора. Отличие применяемой здесь процедуры от процедуры, входящей в III блок, состоит лишь в том, что вместо вычисления матриц Π_{ij} используются соответствующие матрицы, вычисленные III блоком.

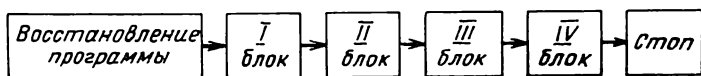


Рис. 6.

3. Подпрограмма формирования ответа о принадлежности вектора к тому или иному классу для каждого из распознаваемых векторов по величине \tilde{L} вычисляет вероятности $\hat{P}(x \in A)$ и $\hat{P}(x \in B)$. Затем по формулам (1.5) и (1.6) определяются ожидаемые значения выигрыша, получаемые при классификации распознаваемого вектора как вектора класса A или B соответственно.

Общая блок-схема программы приведена на рис. 6.

3. ОБ ИСПОЛЬЗОВАНИИ ПРОГРАММЫ РАСПОЗНАВАНИЯ ДЛЯ РЕШЕНИЯ КОНКРЕТНЫХ ЗАДАЧ

Все известные алгоритмы распознавания явно или неявно налагают априорные ограничения на класс решаемых ими задач; иначе говоря, они могут успешно решать далеко не все задачи распознавания, формально поставленные в их терминах [2]. Так, алгоритм, реализуемый данной программой, предполагает, что решающее правило может быть получено в виде функции от статистических оценок точек в подпространствах невысоких размерностей, полученных по обучающей выборке. Следовательно, компоненты решающего правила должны допускать представление их в виде функций малого числа параметров, характеризующих вектор. Это определяет подход к выбору системы параметров при постановке задачи.

Так, например, в задаче различения сейсмограмм землетрясений и шумов, примером неудачно выбранной системы параметров (хотя и содержащей достаточную информацию) является система, в которой каждый параметр — это значение ординаты сейсмограммы для фиксированного (от начала записи) момента времени. Со-

вокупность небольшого числа (2—4) таких ординат мало характеризует кривую. Другое дело, когда в качестве параметров выбираются величины, определяющие кривую в целом, например параметры, характеризующие различные участки спектра мощности кривых.

Аналогичные соображения позволяют учесть некоторые априорные сведения об искомом решающем правиле. Так, если известно, что какая-либо функция от принятых для описания вектора параметров позволяет просто сформулировать правило разделения на классы хотя бы для части векторов, то такую функцию следует добавить к описанию вектора в виде нового, «производного» параметра. При этом система параметров перестает быть независимой (если она была таковой ранее), но это обстоятельство не ухудшает работы алгоритма (см., например, [3]).

Важными параметрами описанного алгоритма являются цены $\{\lambda\}$ (см. § 1). В случае, когда не удастся найти решающее правило, обеспечивающее полное разделение классов, можно модифицировать полученное правило, меняя цены так, что количество ошибок при классификации векторов одного какого-либо класса уменьшится или даже обратится в нуль (обычно за счет увеличения числа ошибок для векторов другого класса). Это особенно важно в задачах отбраковки, когда недопустимо отнесение бракованного изделия к классу годных; такая возможность модификации позволяет отыскать правило, определяющее «группу риска», т. е. сузить область, содержащую все векторы одного класса (и, может быть, часть векторов другого).

Оценка решающего правила по обучающей выборке является смещенной. Для получения несмещенной оценки при отсутствии достаточной экзаменационной выборки используется так называемое «скользящее узнавание», «скользящий контроль» [4], когда поочередно исключаются один или несколько векторов из обучающей выборки, решающее правило находится по оставшимся векторам и проверяется на исключенных; оценка правила получается как средний результат проверки при поочередном исключении всех векторов выборки. Заметим, что в усложненном (и улучшенном) варианте программы процедура «скользящего узнавания» используется уже при создании решающего правила (а не только при его оценке).

Описанная программа применялась в течение ряда лет для решения задач радиоэлектроники, геофизики, медицины, прогнозирования исхода технологических процессов и т. д.

СПИСОК ЛИТЕРАТУРЫ

1. Браиловский В. Л. Об одном методе распознавания объектов, описываемых несколькими параметрами, и возможности его применения. «Автоматика и телемеханика», 1962, № 12.
2. Браиловский В. Л., Лунц А. Л. Формулировка задачи распознавания объектов со многими параметрами и методы ее решения. «Техническая кибернетика». Изд. АН СССР, 1964, № 1.
3. Браиловский В. Л. Алгоритм распознавания объектов со многими параметрами и его приложения. «Техническая кибернетика». Известия АН СССР, 1964, № 2.
4. Лунц А. Л., Браиловский В. Л. Об оценке признаков, получаемых в статистических решающих правилах. «Техническая кибернетика». Известия АН СССР, 1967, № 3.

В. Н. Вапник, Т. Г. Глазкова, А. Я. Червоненкис

Алгоритмы обучения машин распознаванию образов ОП-4, ОП-5, ОП-6, ОП-7

1. ВВЕДЕНИЕ

Алгоритмы ОП-4, ОП-5, ОП-6, ОП-7 так же как и рассмотренные алгоритмы ОП-1, ОП-2, ОП-3, используют метод «обобщенного портрета». Они могут быть реализованы в виде одной программы. При работе каждый из алгоритмов (ОП-5, ОП-6, ОП-7) содержит как составную часть алгоритм с номером, на единицу меньшим.

С помощью алгоритма ОП-4 строится оптимальная разделяющая гиперплоскость, т. е. гиперплоскость, разделяющая множества X и \bar{X} и максимально удаленная от их выпуклых оболочек. С помощью алгоритма ОП-1 строилась оптимальная разделяющая гиперплоскость при условии, что

$$\begin{aligned}(x\psi) &\geq 1, & x \in X, \\ (\bar{x}\psi) &\leq k, & \bar{x} \in \bar{X}, & k < 1.\end{aligned}$$

Алгоритм ОП-4 устанавливает (в случае, если это так), что не существует разделяющей гиперплоскости, проходящей от выпуклых оболочек множеств X и \bar{X} на расстоянии ρ , большем ρ_0 .

С помощью алгоритма ОП-5 определяется подпространство минимальной размерности, в котором множества

могут быть разделены гиперплоскостью так, что расстояния до их выпуклых оболочек больше ρ_0 . Этот алгоритм эвристический и гарантирует достижение только локального минимума.

Иногда оказывается полезным удалить из обучающей последовательности несколько векторов, с тем чтобы получившуюся обучающую последовательность либо разделить на классы в исходном пространстве, либо разделить в подпространстве достаточно малой размерности. Это делается с помощью алгоритма ОП-6.

И, наконец, в случае, когда множества X и \bar{X} не могут быть разделены в исходном пространстве гиперплоскостью, применяется алгоритм ОП-7.

С помощью алгоритма ОП-7 множества X и \bar{X} разделяются кусочно-линейной поверхностью. Однако, в отличие от алгоритма ОП-3 гиперплоскости, образующие эту поверхность, имеют размерность меньше n .

Алгоритм ОП-4 является алгоритмом построения оптимальной гиперплоскости;

алгоритм ОП-5 — алгоритмом минимизации числа признаков;

алгоритм ОП-6 — алгоритмом обучения с исключением векторов из обучающей последовательности;

алгоритм ОП-7 — алгоритмом построения кусочно-линейных поверхностей.

Так же как и алгоритмы ОП-1, ОП-2, ОП-3, перечисленные алгоритмы работают с «очищенной» обучающей последовательностью, т. е. из последовательности изъяты тождественные векторы, принадлежащие различным классам.

Обычно на практике из последовательности исключаются такие векторы $x_i \in X (x_i \subset \bar{X})$, для которых

$$\frac{\min_{x \in X} |x - \bar{x}|}{\max_{x \in \bar{X}} |x - \bar{x}|} < \Pi.$$

Программа алгоритмов ОП-4, ОП-5, ОП-6, ОП-7 может быть написана в двух вариантах: для случая, когда координаты векторов бинарны, и для случая, когда координаты векторов — действительные числа.

Как правило, используется первый вариант программы. При этом способ кодирования векторов тот же, что и для программы ОП-1, ОП-2, ОП-3.

2. ОПИСАНИЕ АЛГОРИТМА ОП-4

Пусть заданы два множества векторов: $X(x_1, \dots, x_a)$ и $\bar{X}(\bar{x}_1, \dots, \bar{x}_b)$.

Рассмотрим множество Z , составленное из всех возможных разностей $x_i - \bar{x}_j$, где $x_i \in X$, а $\bar{x}_j \in \bar{X}$. Пусть z_1, \dots, z_A — элементы этого множества.

Нетрудно видеть, что минимальный по модулю вектор ψ , удовлетворяющий неравенствам

$$(z_i \psi) \geq 1 \quad (i = 1, 2, \dots, A),$$

и число $c = (c_1 + c_2)/2$,

где

$$c_1 = \min_X (x\psi); \quad c_2 = \max_{\bar{X}} (\bar{x}\psi),$$

определяют гиперплоскость $(x\psi) = (c_1 + c_2)/2$, разделяющую множества X и \bar{X} и наиболее от них отдаленную.

Подобно тому, как в алгоритме ОП-1, вектор ψ представим в виде

$$\psi = \sum_{i=1}^A \alpha_i z_i, \quad \alpha_i \geq 0,$$

причем $\alpha_i(1 - (z_i\psi)) = 0$, т. е. в разложение вектора ψ с ненулевым весом входят только те векторы z_i , для которых выполняется равенство $(z_i\psi) = 1$. Коэффициенты разложения α_i могут быть найдены как координаты точки покоя системы уравнений:

$$\dot{\alpha}_i = \begin{cases} 1 - (z_i\psi), & \text{если } \alpha_i \geq 0 \text{ или } (1 - (z_i\psi)) \geq 0, \\ 0 & \text{если } \alpha_i \leq 0 \text{ и } (1 - (z_i\psi)) < 0. \end{cases} \quad (1)$$

Так же как и система (5) для алгоритма ОП-1, эта система состоит из большого числа уравнений. Поэтому решать ее следует не целиком, а по частям. Поиск точки покоя ОП-1 ведется методом «партан» (см. алгоритм ОП-1).

Порядок решения системы (1):

Сначала решается система для нескольких векторов z_i (например, для k). (Начальная группа обучения.) Затем система сокращается: из нее исключаются неинформативные векторы z_i (т. е. те векторы, которые вхо-

дят в разложение вектора ψ с нулевым весом). Далее определяется вектор z , для которого $d = \min_z \{z_i \psi\}$.

Если $d \geq 1 - \kappa$, то вектор ψ построен. Остается теперь для построения гиперплоскости только определить

$$c_1 = \min_x (x\psi); \quad c_2 = \max_{\bar{x}} (\bar{x}\psi).$$

Если же $d < 1 - \kappa$, то вектор z_i добавляется к информативным и вновь решается система (1). Процесс повторяется до тех пор, пока не окажется, что $d \geq 1 - \kappa$.

Так же как и в системе (5) для алгоритма ОП-1, система (1) считается решенной, если $|\alpha_i| < \varepsilon$ ($i = 1, 2, \dots, k$).

Одновременно с решением системы (1) алгоритм ОП-4 вычисляет величину $W = \sum \alpha_i - 1/2(\psi\psi)$. Если в какой-то момент окажется, что

$$W > W_0 = 1/\rho_0^2, \quad (2)$$

то поиск гиперплоскости прекращается. Неравенство (2) означает, что не существует гиперплоскости, разделяющей множества X и \bar{X} и отстоящей от ближайшего из них на расстоянии, большем ρ_0 .

3. ОПИСАНИЕ ПРОГРАММЫ ОП-4

Программа ОП-4 может быть выполнена по аналогии с программой ОП-1. В отличие от последней в эту программу обучающая последовательность вводится двумя массивами: массивом векторов первого класса и массивом векторов второго класса.

В информационной карте к программе указывается:

- 1) точность счета ε (обычно $\varepsilon = 0,1$);
- 2) допустимое отклонение от оптимальной гиперплоскости κ ;
- 3) количество векторов в начальной группе обучения $k = 10 \div 15$;

4) константа $W_0 = 1/\rho_0^2$;

5) размерность пространства бинарных векторов;

6) количество ячеек, занимаемых одним вектором.

Ниже описана блок-схема программы ОП-4.

Блок ОП-4-1

1. Переводит константы из десятичного счисления в двоичное.
2. Передает управление блоку ОП-4-2.

Блок ОП-4-2

1. Подсчитывает число векторов первого и число векторов второго классов.
2. Передает управление блоку ОП-4-3.

Блок ОП-4-3

1. Записывает векторы обучающей последовательности на МБ-3.
2. Записывает векторы обучающей последовательности на МБ-2.
3. Передает управление блоку ОП-4-4.

Блок ОП-4-4

1. Вызывает в группы рабочих ячеек k векторов первого и k векторов второго классов.
2. Записывает нули в ячейки группы R_{α}^I .
3. Передает управление блоку ОП-4-5.

Блок ОП-4-5 («обучение»)

1. Решает систему (ниже этот блок будет подробно рассмотрен).
2. Передает управление блоку ОП-4-6.

Блок ОП-4-6

1. Исключает неинформативные пары векторов из той группы, по которой решалась система (1).
2. Пододвигает (записывает подряд) информативные векторы первого и информативные векторы второго классов.
3. Пододвигает соответствующие коэффициенты разложения вектора.
4. Передает управление блоку ОП-4-7.

Блок ОП-4-7

1. Вычисляет величину $c_1 = \min_x (x\psi)$. Здесь ψ — вектор из группы ячеек R_{ψ}^I (см. блок «обучение», описанный ниже), x — векторы первого класса. Пусть $c_1 = (x^*\psi)$.
2. Вычисляет величину $c_2 = \max_{\bar{x}} (\bar{x}\psi)$. Здесь \bar{x} — векторы второго класса. Пусть $c_2 = (\bar{x}^*\psi)$.
3. Вычисляет разность $d = c_1 - c_2$.
4. Если $d \geq 1 - \kappa$, то определяет порог $c = (c_1 + c_2)/2$, печатает результат решения и выполняет останов.
5. Если $d < 1 - \kappa$, то к группе информативных пар дописывается пара $x^* - \bar{x}^*$, т. е. к векторам первого класса дописывается вектор x^* , а к векторам второго класса — вектор \bar{x}^* .

6. Подсчитывает число информативных пар (пусть их число равно r).

7. Записывает «0» в r -ю ячейку в группе R_{α}^I .

8. Передает управление блоку ОП-4-5.

Подробная схема блока «обучение» ОП-4-5

Блок «обучение» аналогичен блоку алгоритма ОП-1: он также использует рабочие поля R_{α}^I ; R_{α}^{II} ; R_{α}^{III} ; R_{ψ}^I ; R_{ψ}^{II} ; R_{ψ}^{III} ; R_{α} , R_{ψ} .

Блок ОП-4-5-1 (ψ — вне цикла)

1. Вычисляет ψ по информативным парам x_i , \bar{x}_j , записанным в группах информативных векторов первого класса и информативных векторов второго класса, и коэффициентам α , записанным в группе ячеек R_{α}^I . (В начальный момент $\alpha_i = 0$.)

Для этого:

а) вычисляет вектор $\psi_1 = \sum \alpha_i x_i$;

б) вычисляет вектор $\psi_2 = \sum \alpha_i \bar{x}_i$;

в) образует вектор $\psi = \psi_1 - \psi_2$.

Этот вектор записывается в группу ячеек R_{ψ}^I .

2. Передает управление блоку ОП-4-5-2.

Блок ОП-4-5-2 (первая пересылка вне цикла)

1. Содержимое группы ячеек R_{α}^I пересылается в группу ячеек R_{α}^{II} .

2. Содержимое группы ячеек R_{ψ}^I пересылается в группу ячеек R_{ψ}^{II} .

3. Передает управление блоку ОП-4-5-3.

Блок ОП-4-5-3 (градиентный шаг вне цикла)

1. Вычисляет производные $\dot{\alpha}_i$ и записывает их в группу ячеек $R_{\alpha}^{\dot{}}$. Эти производные вычисляются по формулам:

$$\dot{\alpha}_i = \begin{cases} 1 - (x_i \psi) + (\bar{x}_i \psi), & \text{если } \alpha_i \geq 0 \text{ или } 1 - ((x_i - \bar{x}_i) \psi) \geq 0; \\ 0, & \text{если } \alpha_i \leq 0 \text{ и } 1 - ((x_i - \bar{x}_i) \psi) < 0, \end{cases}$$

где ψ — вектор, записанный в группе ячеек R_{ψ}^I .

2. Вычисляет числитель пробного шага $\dot{C} = \sum \dot{\alpha}_i^2$ ($\dot{\alpha}_i$ из группы ячеек $R_{\alpha}^{\dot{}}$).

3. Находит вектор $\dot{\psi} = \sum \dot{\alpha}_i x_i - \sum \dot{\alpha}_i \bar{x}_i$, величины $\dot{\alpha}_i$ хранятся в группе ячеек $R_{\alpha}^{\dot{}}$, $\dot{\psi}$ записывается в группу ячеек $R_{\psi}^{\dot{}}$.

4. Определяет знаменатель пробного шага $3 = (\dot{\psi}\dot{\psi})$, $\dot{\psi}$ хранится в группе ячеек $R_{\dot{\psi}}$.

5. Находит пробный шаг $h_{пр} = 4/3$.

6. Находит истинный шаг $h_{ист} = \min(h_{пр}, |\alpha_i/\dot{\alpha}_i|)$, где отношения $|\alpha_i/\dot{\alpha}_i|$ берутся из группы ячеек R_{α}^I и $R_{\dot{\alpha}}$ для тех i , у которых $\dot{\alpha}_i < 0$.

7. Находит новое приближение вектора $\psi_{н} = \psi + \dot{\psi}h$, где ψ — вектор, записанный в группе ячеек R_{ψ}^I , $\psi_{н}$ записывается в группу ячеек R_{ψ}^I .

8. Вычисляет новые значения $\alpha_{н} = \alpha + \dot{\alpha}h_{ист}$; величины α записаны в группе ячеек R_{α}^I , $\alpha_{н}$ записываются в группу ячеек R_{α}^I .

9. Передает управление блоку ОП-4-5-4.

Блок ОП-4-5-4 (первая пересылка в цикле)

1. Содержимое группы ячеек R_{ψ}^{II} переписывается в группу ячеек R_{ψ}^{III} .

2. Содержимое группы ячеек R_{ψ}^I — в группу ячеек R_{ψ}^{II} .

3. Содержимое группы ячеек R_{α}^{II} — в группу ячеек R_{α}^{III} .

4. Содержимое группы ячеек R_{α}^I — в группу ячеек R_{α}^{II} .

5. Передает управление блоку ОП-4-5-5.

Блок ОП-4-5-5 (градиентный шаг в цикле)

1. Выполняет операции п. 1 в блоке ОП-4-5-3.

2. Проверяет условия $|\dot{\alpha}_i| < \epsilon$. Если условия выполнены, т. е. система решена, то передает управление блоку ОП-4-6. Если же хотя бы для одного i неравенство не выполняется, то передает управление п. 3 данного блока.

3. Выполняет операции пп. 2, 3, 4, 5 блока ОП-4-5-3.

4. Передает управление блоку ОП-4-5-6.

Блок ОП-4-5-6 (проверка условия неделения)

1. Вычисляет величину $W = \Sigma \alpha_i - 1/2 (\psi\psi)$; величины α выбирают из группы ячеек R_{α}^I , а вектор ψ — из группы ячеек R_{ψ}^I .

2. Проверяет неравенство $W > W_0 = 1/\rho_0^2$. Если это неравенство выполнено, то происходит останов по неделению. Если же неравенство не выполнено, то передает управление блоку ОП-4-5-7.

Блок ОП-4-5-7 (овражный шаг)

1. Вычисляет разности $\Delta \alpha_i = \alpha_i^I - \alpha_i^{III}$, величина α_i^I записана в группе ячеек R_{α}^I , α_i^{III} — в группе ячеек R_{α}^{III} , $\Delta \alpha_i$ записывается в группу ячеек $R_{\Delta \alpha}$.

2. Вычисляет вектор $\Delta\psi = \psi^I - \psi^{III}$, где ψ^I — вектор, записанный в группе ячеек R_ψ^I , ψ^{III} — вектор, записанный в группе ячеек R_ψ^{III} , вектор $\Delta\psi$ записывается в группу ячеек R_ψ .

3. Вычисляет числитель \mathcal{C} пробного овразного шага $\mathcal{C} = \Sigma \Delta\alpha_i - (\psi \Delta\psi)$, где величина $\Delta\alpha_i$ записана в группе ячеек R_α , ψ — в группе ячеек R_ψ^I , $\Delta\psi$ — в группе ячеек R_ψ .

4. Меняет знак разностей $\tilde{\Delta\alpha}_i = \text{sign } \mathcal{C} \Delta\alpha_i$, где $\text{sign } \mathcal{C}$ — знак числителя пробного шага; $\tilde{\Delta\alpha}_i$ — записывается в группу ячеек R_α .

5. Меняет знак координат вектора $\Delta\psi: \tilde{\Delta\psi} = \text{sign } \mathcal{C} \Delta\psi$, вектор $\Delta\psi$ записывается в группу ячеек R_ψ ; $\tilde{\Delta\psi}$ — в группу ячеек R_ψ .

6. Выполняет операции пп. 4, 5, 6, 7, 8 блока ОП-4-5-3.

7. Передает управление блоку ОП-4-5-2.

4. ОПИСАНИЕ АЛГОРИТМА ОП-5

Пусть множества X и \bar{X} разделимы гиперплоскостью и $(x\psi) = c$ есть оптимальная разделяющая гиперплоскость.

С помощью алгоритма ОП-5 в пространстве $X^{(n)}$ определяется подпространство минимальной размерности, в котором множества X и \bar{X} могут быть разделены гиперплоскостью так, что расстояние до ближайшего из них не меньше ρ_0 . (Напомним еще раз, что этот алгоритм гарантирует достижение только локального минимума.)

Идея алгоритма состоит в следующем: рассматривается вектор ψ , образующий оптимальную разделяющую гиперплоскость. У этого вектора отыскивается минимальная по модулю, но не равная нулю координата. Если модуль p -й координаты $|\psi_p| > \gamma$ (γ — параметр программы), то считается, что минимизация невозможна.

Если же $|\psi_{p_1}| \leq \gamma$, то на место найденной координаты засылается нуль. Затем определяется следующая минимальная по модулю, но не равная нулю координата. Пусть модуль ее равен $|\psi_{p_2}|$. Если $|\psi_{p_1}| + |\psi_{p_2}| \leq \gamma$, то на место координаты $|\psi_{p_2}|$ также засылается „0“.

Засылка „0“ продолжается до тех пор, пока не окажется, что $\sum_i |\psi_{p_i}| > \gamma$.

Если $\gamma < 1$, то вектор $\tilde{\psi}$, полученный из вектора ψ «зануливанием» координат, и число c образует разделяющую гиперплоскость в подпространстве размерности $(n-t)$. Здесь $(x\psi) = c$ — оптимальная разделяющая гиперплоскость, t — число «зануленных» координат (с нулевыми значениями). Однако найденная гиперплоскость не является оптимальной. Поэтому в полученном подпространстве строится оптимальная гиперплоскость и по ней определяется новое подпространство и т. д.

Процесс минимизации заканчивается, если:

1. Окажется, что в очередном найденном подпространстве не существует разделяющей гиперплоскости, отстоящей от разделяющих множеств на величину $\rho > \rho_0$.

2. Любая координата очередного вектора ψ_i больше γ .

3. Будет найдено подпространство заданной размерности m .

Заметим, что если положить $m=0$, то процесс минимизации закончится при выполнении либо условия 1, либо условия 2. Выбором значения γ можно добиться, чтобы процесс минимизации заканчивался только при выполнении условия 1.

Обычно при использовании программы, реализующей этот алгоритм, число координат сокращается довольно значительно (в 2—5 раз). Поэтому в целях экономии машинного времени целесообразно исключать координаты векторов x , соответствующие «зануленным» координатам вектора ψ . Одновременно исключаются и «зануленные» координаты вектора ψ . Это позволяет проводить вычисления с векторами меньшей размерности.

5. ОПИСАНИЕ ПРОГРАММЫ ОП-5

Программа ОП-5 использует как составную часть программу ОП-4.

К информационной карте этой программы добавляется информация о константах:

1) константа γ (обычно 0,8);

2) желаемая размерность подпространства m (в случае минимизации размерности пространства $m=0$).

В программу ОП-4 вносятся поправки: в блоке ОП-4-7 вместо команды «Останов» следует «Передать управление блоку ОП-5-1».

Рассмотрим блок-схему программы ОП-5.

Блок ОП-5-1

1. По характеристическому вектору (см. блок ОП-5-4) и вектору $\hat{\psi}$ из группы ячеек R_{ψ}^I печатает вектор $\hat{\psi}$ в исходном n -мерном пространстве; этот вектор ψ образуется из вектора ψ добавлением соответствующих нулевых координат. Номера нулевых координат соответствуют номерам нулевых разрядов характеристического вектора.
2. Передает управление блоку ОП-5-2

Блок ОП-5-2

1. Сравнивает размерность полученного пространства n_i с желаемой размерностью m . Если $n_i \leq m$, то выполняется команда «Останов». Если же $n_i < m$, то передает управление блоку ОП-5-3.

Блок ОП-5-3

1. Записывает ноль в ячейку Y_0 .
2. Отыскивает минимальную по модулю и не равную нулю координату вектора ψ подгруппы R_{ψ}^I .
3. Суммирует величину, равную модулю этой координаты $|\psi_{p_1}|$ с содержимым ячейки Y_0 . Если полученная величина меньше γ , то вместо этой координаты засылает «0» и передает управление п. 2 блока ОП-5-3. Если же эта величина больше γ , то сравнивает величину модуля найденной координаты с содержимым ячейки Y_0 ; при равенстве этих величин произойдет останов, так как дальнейшая минимизация размерности пространства невозможна. Если же эти величины не равны, то передает управление блоку ОП-5-4.

Блок ОП-5-4

1. Перестраивает характеристический вектор, т. е. бинарный вектор размерности n исходного пространства, где нулевая i -я координата характеристического вектора означает, что i -я координата вектора ψ равна «0», «1» — что i -я координата не равна «0». Число единиц в характеристическом векторе равно размерности найденного подпространства. Перед началом минимизации характеристический вектор состоит из единиц.
2. Передает управление блоку ОП-5-5.

Блок ОП-5-5

1. Сокращает размерность векторов множества X , исключая из векторов обучающей последовательности те координаты, которым соответствует «0» в векторе $\psi (R_{\psi}^I)$.
2. Сокращает размерность вектора ψ , исключая нулевые координаты.
3. Передает управление блоку ОП-4-4.

6. ОПИСАНИЕ АЛГОРИТМА ОП-6

Алгоритм ОП-6 применяется в тех случаях, когда в подпространстве заданной размерности m множества X и \bar{X} вообще не могут быть разделены гиперплоскостью;

m может быть равно исходной размерности n , а может быть и меньше нее (см. описание программы ОП-5). После исключения из обучающей последовательности некоторых векторов оставшееся множество может быть разделено гиперплоскостью в подпространстве размерности m . Алгоритм ОП-6 включает как составную часть алгоритм ОП-5.

В случае, если в подпространстве размерности m не может быть построена разделяющая гиперплоскость, произойдет останов по условию 1 или 2 (см. описание программы ОП-5).

С помощью алгоритма ОП-6 выясняется, какой вектор должен быть исключен из обучающей последовательности, с тем чтобы построение требуемой гиперплоскости могло быть продолжено.

Для этого каждой паре x_i и \bar{x}_i информативных векторов ставятся в соответствие два числа: $\alpha_i(1 - x_i\psi)$ и $\alpha_i(1 - (\bar{x}_i\psi))$, где α_i — коэффициент, с которым входит вектор $x_i - \bar{x}_i$ в разложение ψ .

Вектор x_i может одновременно входить в несколько информативных пар (аналогично и вектор \bar{x}_i). Поставим

в соответствие каждому вектору x_i число $(1 - (x_i\psi)) \sum_{i=1}^p \alpha_i$,

где p — число информативных пар, в которых встречается вектор x_i , т. е. суммирование ведется по тем информативным парам, в которых встречается этот вектор.

Аналогично каждому вектору \bar{x}_i ставится в соответствие число $(1 - (\bar{x}_i\psi)) \sum_{i=1}^p \alpha_i$. Здесь суммирование ведется

по всем информативным парам, где встречается вектор \bar{x}_i .

Найдем максимум полученных чисел. Из обучающей последовательности исключим вектор, соответствующий этому максимуму. После удаления вектора снова делается попытка в зависимости от значений информационных констант либо разделить обучающую последовательность, либо уменьшить размерность пространства.

7. ОПИСАНИЕ ПРОГРАММЫ ОП-6

В программу ОП-5 внесены следующие изменения. В блоке ОП-4-5-6 вместо команды «Останов» выполняется команда «Передать управление блоку ОП-6-1».

В блоке ОП-5-2 команда «Останов» заменяется командой «Передать управление блоку ОП-6-1».

Информационная карта к программе ОП-6 та же, что и к программе ОП-5.

Блок ОП-6-1

1. Для каждого информативного вектора первого класса x_i находит величину $(1 - (x_i \psi)) \sum_{i=1}^p \alpha_i$ (суммирование ведется по всем i , которым соответствует вектор x_i).

2. Отыскивает вектор x^* с максимальным значением величины $(1 - (x^* \psi)) \sum \alpha_i$.

3. Для каждого информативного вектора второго класса аналогично находит значение $(1 - (\bar{x}_i \psi) \sum \alpha_i$.

4. Отыскивает вектор \bar{x}^* с максимальным значением $(1 - (x^* \psi)) \sum \alpha_i$.

5. Находит вектор \hat{x} , соответствующий максимуму этих двух чисел $(1 - (x^* \psi)) \sum \alpha_i$, $(1 - (\bar{x}^* \psi)) \sum \alpha_i$.

6. Передает управление блоку ОП-6-2.

Блок ОП-6-2

1. Исключает из обучающей последовательности найденный вектор.

2. Добавляет к счетчику числа информативных векторов единицу.

3. Передает управление блоку ОП-4-4.

8. ОПИСАНИЕ АЛГОРИТМА ОП-7

Алгоритм ОП-7 содержит в себе как составную часть алгоритм ОП-6, т. е. этот алгоритм в определенные моменты может обращаться либо к алгоритму построения оптимальной разделяющей гиперплоскости ОП-4, либо к алгоритму минимизации размерности пространства признаков ОП-5, либо к алгоритму исключения из обучающей последовательности определенных векторов ОП-6. Это позволяет алгоритму ОП-7 в пространстве признаков построить кусочно-линейные поверхности, разделяющие множества обучающей последовательности и удовлетворяющие некоторому критерию.

Идея алгоритма ОП-7 состоит в следующем. Пусть ставится задача обучения классификации, например, на два класса. Для каждого класса конструируется некото-

рое количество (не обязательно одинаковое) признаков, удовлетворяющих условиям:

1. Каждый признак первого (соответственно второго) класса классифицирует обучающую последовательность, делая только ошибки первого рода, т. е. он может относить некоторые векторы первого (соответственно второго) класса ко второму (соответственно к первому) классу, но никогда не относит вектор, не принадлежащий первому (соответственно второму) классу, к первому классу (соответственно второму). Такие признаки называются *достаточными*. Обозначим i -й достаточный признак первого класса через $\pi_i(x)$, а j -й достаточный признак второго класса — через $\pi_j(x)$.

2. Совокупность признаков $\pi(x) = \{\pi_i(x)\}$ или $\bar{\pi}(x) = \{\pi_j(x)\}$ такова, что для любого вектора $x(\bar{x})$ обучающей последовательности, принадлежащего первому (второму) классу, существует хотя бы один признак, относящий этот вектор к первому классу (соответственно второму). А согласно условию 1 ни для одного из векторов обучающей последовательности не существует признака, ошибочно относящего его к другому классу.

Совокупность таких признаков определяет следующее решающее правило: для каждого вектора x подсчитывается число $v_1(x)$ признаков, относящих его к первому классу, и число признаков $v_2(x)$, относящих его ко второму классу, т. е. для каждого вектора x подсчитываются величины $v_1(x)$ и $v_2(x)$.

Вектор x относится к первому классу, если $v_1(x) - v_2(x) > 0$, и ко второму классу, если $v_1(x) - v_2(x) < 0$. Если же $v_1(x) = v_2(x)$, то алгоритм отказывается классифицировать данный вектор.

В качестве признака в алгоритме ОП-7 выбирается положение вектора относительно гиперплоскости в подпространстве размерности не выше m пространства E^n (m — константа алгоритма).

Признаки для каждого класса строятся последовательно. Процессы построения совокупности достаточных признаков для первого и второго классов аналогичны. Поэтому мы опишем его только для первого класса.

Определяется такая оптимальная гиперплоскость в подпространстве размерности не выше m , что по одну сторону от нее лежит максимальное количество векторов первого класса и нет ни одного вектора второго класса. Вектор ψ и число c этой гиперплоскости запоминаются.

Затем из обучающей последовательности исключаются все информативные векторы первого класса и снова определяется гиперплоскость в подпространстве размерности не выше m , удовлетворяющая тем же условиям, что для первого признака, но уже по уменьшенной обучающей последовательности.

Признаки строятся до тех пор, пока не окажется, что при построении очередного признака было исключено менее A векторов (A — параметр алгоритма).

9. ОПИСАНИЕ ПРОГРАММЫ ОП-7

Программа ОП-7 строит совокупность достаточных признаков первого класса. Если стоит задача обучения классификации на k классов, то, назначая последовательно j -й класс первым, а все остальные — вторым ($j = 1, 2, \dots, k$), строятся совокупности достаточных признаков для всех k классов.

Программа ОП-7 использует программу ОП-6, в которую внесены следующие изменения: в блоке ОП-6-2 исключение векторов производится только из множества векторов первого класса. Для этого в блоке ОП-6-1 вместо п. 3 выполняется команда «Передать управление блоку ОП-6-2». В блоке ОП-5-2 вместо п. 2 («Останов») следует п. 2: «Передать управление блоку ОП-7-1».

В информационной карте к программе ОП-7 добавляются сведения о константе A .

Описание блок-схемы ОП-7 приведено ниже.

Блок ОП-7-1

1. Сравнивает содержимое счетчика $Sч$ числа исключенных векторов с эталоном A . Если $\{Sч\} \leq A$, то выполняет останов. Если же $\{Sч\} > A$, то передает управление блоку ОП-7-2.

Блок ОП-7-2

1. Записывает «0» в ячейку $Sч$.
2. Восстанавливает характеристический вектор (во всех разрядах единицы).
3. Передает управление блоку ОП-7-3.

Блок ОП-7-3

1. Переписывает векторы обучающей последовательности с МБ-2 на МБ-3.

2. Исключает из группы векторов первого класса, записанных на МБ-3, все информативные векторы первого класса. Оставшиеся векторы записывает подряд.

3. Передает управление блоку ОП-4-4.

Для классификации векторов, не участвовавших в обучении, используется программа, которая применяет для классификации полученное в процессе обучения решающее правило.

Если решающее правило получено с помощью программ ОП-4, ОП-5 и ОП-6, то программа «экзамен» та же, что и для программы ОП-1, ОП-2.

Для решающих правил, полученных с помощью программы ОП-4, ОП-6 (при $n = m$), может быть применена программа «скользящий контроль».

В случае же, если решающее правило было получено с помощью программы ОП-7, процедура «экзамен» состоит в следующем: с помощью программы ОП-7 определяется совокупность достаточных признаков для k классов (в случае, если классификация должна быть проведена на k классов). Во время «экзамена» считается, сколько признаков каждого класса содержится в предъявленном векторе x . Этот вектор относится к тому классу, число признаков которого максимально.

**З. С. Баталова, Е. Ф. Куренков, Ю. И. Неймарк,
Н. Д. Образцова**

Некоторые алгоритмы медицинской диагностики и прогнозирования

Рассмотрим некоторые алгоритмы и программы распознавания образов и отбора существенных признаков, реализованные на электронных вычислительных машинах М-220 и БЭСМ-3М. Эти алгоритмы разрабатывались применительно к задачам медицинской диагностики, прогнозирования исхода заболеваний и оперативных вмешательств, а также для оптимизации выбора метода лечения.

Все приведенные алгоритмы основаны на предположении, что при достаточной полноте описания заболевания близким признакам болезни соответствуют близкие диагнозы или прогнозы. Эти алгоритмы используются при наличии обучающего материала (статистических

данных истории болезни), закодированного по определенным правилам.

Ниже дается описание алгоритма диагностики и прогнозирования по общей близости признаков, который используется как для принятия решений, так и для анализа статистического материала. В качестве меры сходства объектов (больных) принята такая характеристика, в которой учитывается не только близость в значениях каких-либо признаков, но и придается определенный вес самим признакам.

Значения весов признаков находятся путем построения так называемой *раздвигающей метрики* на основе исходного статистического материала. По виду этой метрики можно судить об относительной значимости отдельных признаков или их линейных комбинаций для распознавания образов и указать как наиболее важные признаки, так и несущественные. В раздвигающей метрике для всех точек пространства признаков их весовые коэффициенты имеют постоянное значение. В работе приводится ряд алгоритмов построения раздвигающей метрики, позволяющих в некоторых случаях уменьшить число признаков без значительного ухудшения результатов распознавания. Обычно эти алгоритмы применяются для отыскания весов признаков в задачах диагностики различных заболеваний.

В случае прогнозирования исходов заболеваний или оперативных вмешательств, когда оценка состояния больного носит непрерывный характер, для определения весовых коэффициентов признаков чаще всего используется алгоритм, основанный на предположении о локальной линейной зависимости прогнозируемого параметра от расстояния между точками в пространстве признаков. С помощью этого алгоритма находятся веса признаков, принимающие различные значения в зависимости от положения точек в пространстве признаков.

Некоторые результаты использования предлагаемых алгоритмов для решения перечисленных выше задач клинической медицины содержатся в работах [1—3].

КОДИРОВАНИЕ СТАТИСТИЧЕСКОГО МАТЕРИАЛА

В качестве признаков, характеризующих состояние больного, могут быть взяты: объективные и анамнестические данные, результаты лабораторных исследований,

данные рентгеноскопии и другие показатели. Условно эти признаки можно разбить на две группы.

К первой группе отнесем признаки, которые характеризуются количественными показателями (например, возраст, температура, частота пульса, кровяное давление, жизненная емкость легких, минутный объем дыхания и др.). Вторую группу признаков составляют сведения о больном, выраженные в медицинских терминах и понятиях, которые не имеют цифровых значений. К этой группе можно отнести различные симптомы заболеваний (наличие и характер болей, слабость, одышка), результаты рентгенологического исследования, заболевания в семье, анамнез и др.

Для кодирования признаков первой группы область изменения каждого признака разбивается на несколько интервалов (градаций) в зависимости от требуемой степени детализации описания признака и поставленной задачи. Интервалы изменения одного и того же признака могут быть различными. Так, в случае диагностики гипертрофии левого желудочка сердца [1] по возрасту выделены следующие интервалы: до 30 лет, 31—40, 41—50, более 50 лет. При дифференциальной диагностике ревматизма и тиреотоксикоза [2] границы интервалов были выбраны: до 20 лет, 20—29, 30—39, более 40 лет.

Признаки, входящие во вторую группу, подразделяются на градации в соответствии с нарастанием симптома или его монотонным изменением. Например, признаку «одышка» соответствуют градации: нет одышки, одышка при физической нагрузке, одышка покоя.

При кодировании признаков обеих групп градациям в порядке их следования присваиваются кодовые числа 0, 1, 2, ..., 7, так что общее число градаций не превышает восьми. Если признак имеет только две градации (например, пол), то им присваиваются любые различные целые числа от 0 до 7. В некоторых случаях, когда признак имеет меньше восьми градаций, целесообразно увеличить разрыв между градациями, пропустив некоторые кодовые числа.

Обозначим через π признак, указывающий диагноз заболевания больного или оценку его состояния. Кодирование этого признака также производится в соответствии с тяжестью заболевания: монотонному изменению состояния больного должно соответствовать монотонное нарастание или уменьшение кодовых чисел 0, 1, ..., 7,

Например, для оценки степени послеоперационной дыхательной недостаточности при резекции легких [3] приняты следующие градации:

0 — дыхательная недостаточность практически отсутствует;

2 — небольшая, с удовлетворительным состоянием больного;

4 — значительная, с тяжелым состоянием;

7 — летальный исход по причине прогрессирующей дыхательной недостаточности.

В запоминающем устройстве машины М-20 кодовые числа признаков x_1, x_2, \dots, x_N и признаки π записываются в восьмеричной системе счисления в ячейки $x+k$ ($k=1, 2, \dots, K$). При этом принято следующее распределение разрядов:

| | | | | | | |
|--------------|---------------|----------|----------|----------|-----|----------|
| Ячейка $x+1$ | Номер разряда | 45—43 | 42—40 | 39—37 | ... | 3—1 |
| | Содержимое | π | x_1 | x_2 | ... | x_{14} |
| Ячейка $x+2$ | Номер разряда | 45—43 | 42—40 | 39—37 | ... | 3—1 |
| | Содержимое | x_{15} | x_{16} | x_{17} | ... | x_{29} |

При наличии нескольких прогнозируемых признаков (рецидивы, осложнения, ближайший и отдаленный исход операции и др.) их коды ставятся после кодов признаков. При этом «экзамен» производится последовательно по всем признакам, причем в каждом случае осуществляется засылка кода соответствующего признака в 43—45 разряды ячейки $x+1$. В дальнейшем через K обозначено число ячеек, необходимых для записи кодов всех признаков и прогнозируемых параметров.

ДИАГНОСТИКА И ПРОГНОЗИРОВАНИЕ ЗАБОЛЕВАНИЙ ПО ОБЩЕЙ БЛИЗОСТИ ПРИЗНАКОВ

Пусть в N -мерном пространстве признаков E_N задана обучающая последовательность точек $\{x^i(x_1, x_2, \dots, \dots, x_N)\}$ ($i=1, 2, \dots, J$), для каждой из которых опреде-

лено значение признака π_i (диагноз или прогноз заболевания). Под расстоянием между точками $x(x_1, x_2, \dots, x_N)$ и $y(y_1, y_2, \dots, y_N)$ в пространстве E_N будем понимать величину

$$\rho(x, y) = \left[\sum_{n=1}^N g_n (x_n - y_n)^2 \right]^{1/2}, \quad (1)$$

где g_1, g_2, \dots, g_N — заданные весовые коэффициенты, удовлетворяющие условию нормировки

$$g_1 + g_2 + \dots + g_N = N. \quad (2)$$

Величина признака π для любой точки $y(y_1, y_2, \dots, y_N)$ пространства E_N определяется следующим образом. Находится ближайшая к точке y точка x^{i*} обучающей последовательности и строится эллипсоид вида

$$\sum_{n=1}^N g_n (x_n - y_n)^2 = [\rho(y, x^{i*}) + \varepsilon]^2 \quad (3)$$

с центром в точке y и полуосями

$$[\rho(y, x^{i*}) + \varepsilon] / \sqrt{g_1}, [\rho(y, x^{i*}) + \varepsilon] / \sqrt{g_2}, \dots, \\ [\rho(y, x^{i*}) + \varepsilon] / \sqrt{g_N},$$

где ε — параметр алгоритма.

Затем вычисляется среднее значение признака по формуле

$$\bar{\pi} = \frac{1}{m} \sum_{S=1}^m \pi_S, \quad (4)$$

(суммирование производится по всем точкам обучающей последовательности, попавшим в этот эллипсоид). При дифференциальной диагностике различных заболеваний, когда признак π может принимать только целые значения, значение π для точки y выбирается равным ближайшей целой части $\bar{\pi}$. При этом ответ для точки y будем считать достоверным, если в эллипсоид (3) попадают точки обучающей последовательности с одинако-

вым значением признака π и если выполняется неравенство

$$\rho(y, x^{i*}) < \rho^*.$$

Ответ будем считать менее достоверным в следующих случаях:

а) в эллипсоид (3) попадают точки с одинаковым значением π , но при этом выполняется соотношение

$$\rho^* \leq \rho(y, x^{i*}) \leq R^*;$$

б) в эллипсоид (3) попадают точки с различными значениями π , но при этом дробная часть $\{\bar{\pi}\} \leq \delta$ и $\rho(y, x^{i*}) < \rho^*$. Случай, когда выполняется неравенство $\{\bar{\pi}\} > \delta$ или неравенство $\rho(y, x^{i*}) > R^*$, рассматривается как *отказ в ответе*. При прогнозировании исходов заболеваний или оперативных вмешательств, когда признак π имеет непрерывный характер, значение π для точки y принимается равным $\bar{\pi}$. При этом достоверность ответа оценивается только по величине $\rho(y, x^{i*})$. Ответ считается достоверным при $\rho(y, x^{i*}) < \rho^*$, менее достоверным при $\rho^* \leq \rho(y, x^{i*}) \leq R^*$, отказ в ответе дается при $\rho(y, x^{i*}) > R^*$.

Введенные здесь величины δ , ρ^* и R^* являются параметрами алгоритма *). Они выбираются экспериментально на основе результатов «экзамена», проводимого для каждой точки обучающей последовательности на базе всего статистического материала (за исключением этой точки). При этом ответ для точки x^i обучающей последовательности считается правильным, если выполняется неравенство.

$$\Delta\pi = |\pi_i - \pi_{i*}| \leq \delta^*,$$

где π_{i*} — фактическое значение признака π для точки x^i ; π_i — величина признака π , полученная с помощью описанного алгоритма. При выполнении соотношения $\Delta\pi > \delta^*$ ответ считается неправильным.

В задачах диагностики и прогнозирования значения величины δ^* могут сильно различаться. Так, при прогнозировании дыхательной недостаточности, когда величина π , как указывалось выше, принимает значения 0, 2, 4,

) При достаточном объеме исходного статистического материала параметр R^ может быть задан достаточно малой величиной.

7, разница в ответе на 1—1,5 градации вполне допустима, в то время как при диагностике двух залобеваний, когда величина π принимает значения 0 и 1, ответы, для которых $0,8 < \Delta\pi \leq 1$, можно считать ошибочными. Обычно в задачах диагностики принято $\delta^* = \delta \leq 0,2$, а в задачах прогнозирования $\delta^* \leq 1,5$.

Проведение «экзамена» поочередно для каждого больного при значениях $g_n = 1$ ($n = 1, 2, \dots, N$) позволяет также выяснить возможность получения достаточно достоверного ответа по выбранному материалу. При этом если количество ошибок при малых значениях $\rho(y, x^{i*})$ превосходит ожидаемое в данной задаче количество ошибок, то это означает, что либо выбранных признаков недостаточно для получения достоверного ответа, либо в статистическом материале имеется слишком много погрешностей. Если же ошибки в ответе и большинство правильных ответов получаются при больших $\rho(y, x^{i*})$, то это говорит о необходимости увеличения объема статистического материала.

При удовлетворительном результате «экзамена» ставится задача отыскания весовых коэффициентов g_1, g_2, \dots, g_N и выделения существенных признаков, решение этой задачи дается ниже.

Блок-схема программы, реализующей алгоритм диагностики и прогнозирования по общей близости признаков, приведена на рис. 1. Здесь через y^l ($l = 1, 2, 3, \dots, L$) обозначены точки экзаменационной последовательности, через x^i ($i = 1, 2, \dots, J$) — точки обучающей последовательности.

Сначала для каждой точки y^l по формуле (1) вычисляются величины $\rho_i(y^l, x^i)$ и находится значение $\rho = \min_i \rho_i$. При этом координаты x_n и y_n ($n = 1, 2, \dots, N$) точек y^l и x^i преобразуются в обычные двоичные числа, удобные для использования в ЭВМ М-220. Затем находятся точки x^i , для которых $\rho(y^l, x^i) \leq \rho + \varepsilon$, и суммируются значения параметра π_i , соответствующие этим точкам. Далее по формуле (4) вычисляется величина $\bar{\pi}$ и находится ближайшее к $\bar{\pi}$ целое число.

На печать выдаются: l — номер экзаменационной точки y^l , величины π_l и $\bar{\pi}_l$, номера i точек обучающей последовательности, попавших в эллипсоид (3), и соответствующие им значения π_i и $\rho(y^l, x^i)$.

Для использования программы задаются следующие данные: L и J — число точек в экзаменационной и обучающей последовательностях, N — количество признаков, K — число ячеек, необходимых для задания информации об одном больном, параметр ϵ , g_1, g_2, \dots, g_N — весовые коэффициенты (при $g_n=1$ коэффициенты не за-

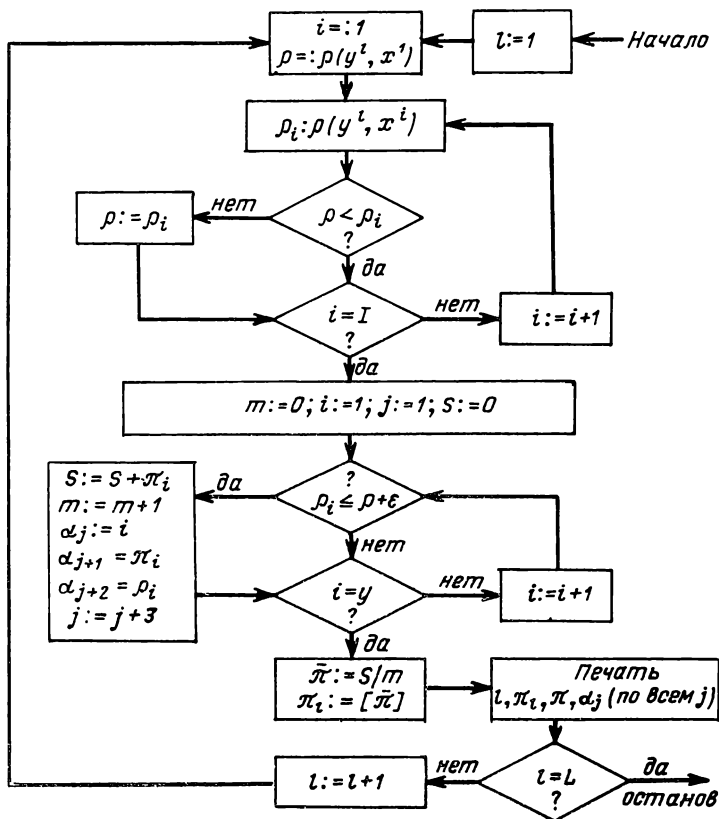


Рис. 1.

даются), A_x и A_y — начальные адреса массивов ячеек МОЗУ, содержащих информацию об экзаменационной и обучающей последовательностях. Программа вместе с необходимыми для ее работы постоянными и формирующей частью занимает 70 ячеек. Для записи экзаменационной и обучающей последовательностей отводится 2470 ячеек МОЗУ.

Пусть в N -мерном пространстве признаков E_N заданы два множества точек $X(x^1, x^2, \dots, x^J)$ и $Y(y^1, y^2, \dots, y^J)$. Раздвигающей называется метрика вида

$$\rho(x, y) = \left[\sum_{m=1}^N \sum_{n=1}^N g_{mn} (x_m - y_m)(x_n - y_n) \right]^{1/2}, \quad (5)$$

в которой при выполнении условия нормирования

$$g_{11} + g_{22} + \dots + g_{NN} = N \quad (6)$$

и неотрицательности квадратичной формы

$$\rho^2(x, y) = \sum_{m=1}^N \sum_{n=1}^N g_{mn} (x_m - y_m)(x_n - y_n) \quad (7)$$

расстояние между множествами X и Y :

$$\rho(X, Y) = \min_{i, j} \rho(x^i, y^j)$$

имеет наибольшее значение.

В этом параграфе приводятся два алгоритма отыскания диагональных коэффициентов g_1, g_2, \dots, g_N .

Использование первого, наиболее простого алгоритма, не всегда позволяет найти раздвигающую диагональную метрику, однако для многих практических задач диагностики приводит к получению достаточно хороших результатов. С помощью второго алгоритма находятся диагональные коэффициенты, близкие к предельным значениям коэффициентов раздвигающей метрики.

С помощью этих алгоритмов можно выяснить вопрос о влиянии отдельных признаков на распознавание и указать как наиболее важные, так и несущественные признаки, что, в конечном счете, позволяет уменьшить размерность исходного пространства признаков. Определение искомых весовых коэффициентов g_1, g_2, \dots, g_N производится последовательно, исходя из условия, что на каждом шаге их изменения не происходит уменьшения расстояния между множествами X и Y .

Рассмотрим построение первого алгоритма.

Пусть g_1, g_2, \dots, g_N — положительные значения весовых коэффициентов на p -м шаге, удовлетворяющие усло-

вию (2), и расстояние ^{*)} между множествами X и Y при этих значениях равно

$$\rho_p = \min_{i, j} \rho(x^i, y^j) = \rho(x^*, y^*).$$

Поиск раздвигающей метрики на $(p+1)$ -м шаге состоит в выполнении следующих операций.

Находятся величины

$$z_{n^*} = \max_n (x_n^* - y_n^*), \quad z_{n_1} = \min_n |x_n^* - y_n^*|.$$

Производится изменение весовых коэффициентов по формуле

$$g'_{n^*} = g_{n^*} + \delta g, \quad g'_{n_1} = g_{n_1} - \delta g, \quad (8)$$

где

$$\delta g = \min(g_{n_1}, 2^{-k}) \quad (k = 2, 3, \dots); \quad (9)$$

остальные весовые коэффициенты не меняются.

Вычисляется расстояние между множествами X и Y с новыми весовыми коэффициентами:

$$\rho_{p+1} = \min_{i, j} \rho(x^i, y^j).$$

При $\rho_{p+1} > \rho_p$ полученные в п. 2 значения $g'_1, g'_2, \dots, \dots, g'_N$ принимаются в качестве весовых коэффициентов на $(p+1)$ -м шаге. Если $\rho_{p+1} < \rho_p$, то значение k , входящее в равенство (9), увеличивается на единицу и производится пересчет весовых коэффициентов по формуле (8). Если с увеличением k окажется выполненным неравенство $\delta g < \delta$ (δ — некоторая малая положительная величина), то находится номер n_2 , для которого

$$z_{n_2} = \min_{n \neq n_1} |x_n^* - y_n^*|,$$

и осуществляется изменением весовых коэффициентов согласно формуле (8), где значение n_1 заменено на n_2 . Если вновь выполняется неравенство $\delta g < \delta$, то находится номер n_3 , для которого

$$z_{n_3} = \min_{n \neq n_1, n_2} |x_n^* - y_n^*|.$$

Процесс повторяется до тех пор, пока не будут найдены такие номер n_l и величина $\delta g > \delta$, при которых

^{*)} В качестве начальных значений весовых коэффициентов выбираются $g_n = 1$ при всех $n = 1, 2, \dots, N$.

$\rho_{p+1} \geq \rho_p$. Если номера максимальной и минимальной величин z_{n^*} и z_{n_l} совпадут, т. е. если $n^* = n_l$ ($l=1, 2, \dots$),

то процесс отыскания весовых коэффициентов g_1, g_2, \dots, g_N заканчивается.

Очевидно, что на каждом шаге изменения весовых коэффициентов необходимо найти величину расстояния между множествами X и Y . Для этого нужно вычислить произведение $J_x \times J_y$ расстояний между точками этих множеств и среди них найти наименьшее. Поскольку при малых изменениях весовых коэффициентов величина $\rho(X, Y)$ меняется незначительно, то с целью уменьшения объема вычислений в программе, реализующей описанный алгоритм, производится сначала отыскание M пар точек x^{i_m} и y^{j_m} , наиболее близко расположенных друг к другу, так что

$$\rho_1(x^{i_1}, y^{j_1}) \leq \rho_2(x^{i_2}, y^{j_2}) \leq \dots \leq \rho_M(x^{i_M}, y^{j_M}).$$

В качестве расстояния между множествами X и Y на последующих Q циклах изменения весовых коэффициентов принимается величина

$$\rho^* = \min_m \rho_m(x^{i_m}, y^{j_m}),$$

где x^{i_m} и y^{j_m} ($m=1, 2, \dots, M$ — отобранные точки).

Изменение весовых коэффициентов при учете только этих точек осуществляется в соответствии с блок-схемой, представленной на рис. 2 (блок А). В результате выполнения указанных здесь операций находятся такие значения весовых коэффициентов g_1, g_2, \dots, g_N , при которых величина ρ^* не уменьшается. После последовательного изменения значений g_1, g_2, \dots, g_N Q раз, производится поиск расстояния между множествами X и Y путем перебора всех точек этих множеств. Если при этом расстояние между множествами X и Y не уменьшилось, т. е. если $\rho_1 \geq \rho^*$, то вектор $g(g_1, g_2, \dots, g_N)$ и величина ρ_1 запоминаются (на рис. 2 этот блок обозначается $\rho_0 := \rho_1$; $g_0 := g$) и производится переход к следующему шагу отыскания весовых коэффициентов.

Неравенство $\rho_1 < \rho^*$ может выполняться при больших значениях Q или δ_g . Поэтому в случаях, когда $\rho_1 < \rho^*$, при $Q > 1$ уменьшается на единицу число Q , а при $Q = 1$ уменьшается вдвое величина δ_g . При этом каждый раз осуществляется восстановление вектора $g(g_1, g_2, \dots, g_N)$

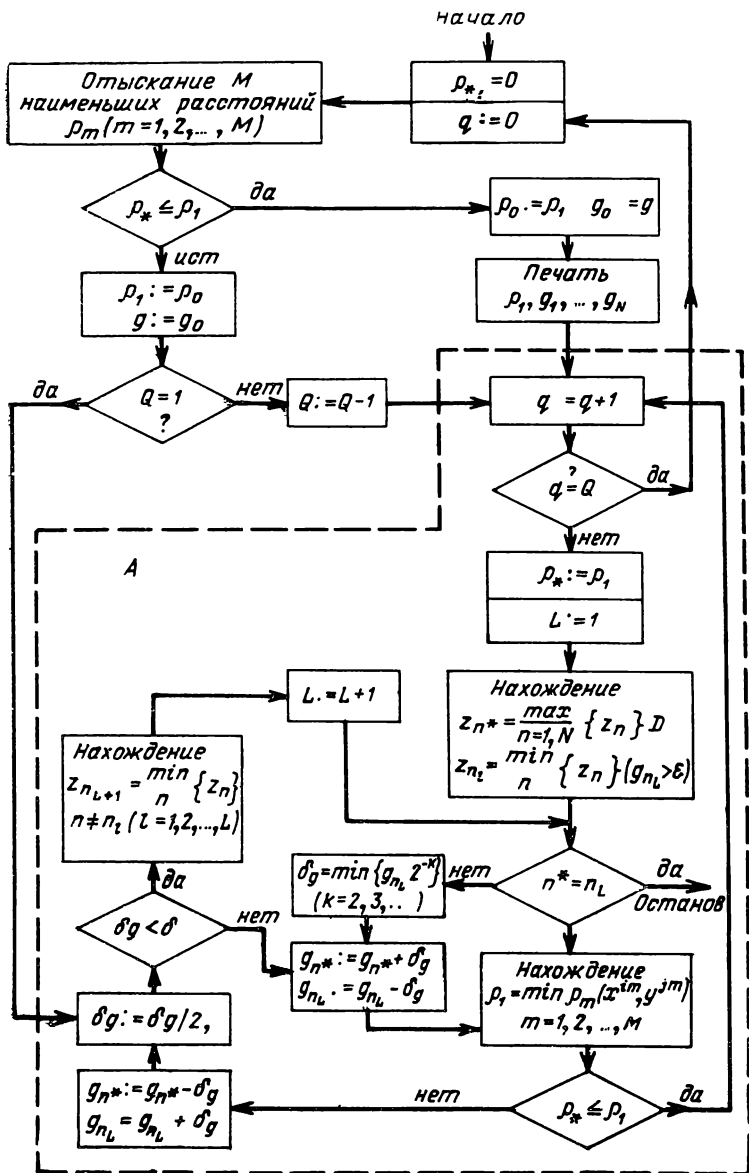


Рис. 2.

и величины ρ_1 (на рис. 2 блок восстановления обозначен $\rho_1 : = \rho_0$; $g : = g_0$). Поиск весовых коэффициентов продолжается начиная с этих значений.

Программа этого алгоритма реализована на машине БЭСМ-3М; она занимает 280 ячеек. Для работы программы задаются величины J_x , J_y , N , K и Q . Используется МОЗУ, в ячейки которого записываются величины $z_{mn}^2 = (x_n^i - y_n^j)^2$ ($n = 1, 2, \dots, N$, $m = 1, 2, \dots, M$), соответствующие отобраным точкам. Поэтому величина M определяется условием $M \leq [4090/N]$. Для размещения статистического материала отводится 1 200 ячеек МОЗУ.

Рассмотрим построение второго алгоритма.

В этом случае отыскание на $(p+1)$ -м шаге изменений $\delta g_1, \delta g_2, \dots, \delta g_N$ весовых коэффициентов g_1, g_2, \dots, g_N , при которых они остаются неотрицательными и при которых не происходит уменьшения расстояния между множествами X и Y , осуществляется следующим образом.

Выбирается M пар точек $(x^{i_1}, y^{j_1}), (x^{i_2}, y^{j_2}), \dots, (x^{i_M}, y^{j_M})$ из множеств X и Y , наиболее близко расположенных друг к другу, так чтобы

$$\rho_1^2(x^{i_1}, y^{j_1}) \leq \rho_2^2(x^{i_2}, y^{j_2}) \leq \dots \leq \rho_M^2(x^{i_M}, y^{j_M}),$$

$$\rho_1^2 = \rho^2(X, Y) = \max_{i,j} \rho^2(x^i, y^j).$$

Приращение квадрата расстояния между этими точками равно

$$\delta \rho^2(x^{i_m}, y^{j_m}) = \sum_{n=1}^N \delta g_n (x_n^{i_m} - y_n^{j_m})^2.$$

Определим величины $\delta g_1, \delta g_2, \dots, \delta g_N$ так, чтобы имели место соотношения *)

$$\delta g_1 + \delta g_2 + \dots + \delta g_N = 0; \quad (10)$$

$$g_n + \delta g_n \geq 0 \quad (n = 1, 2, \dots, N); \quad (11)$$

$$\sum_{n=1}^N \delta g_n (x_n^{i_m} - y_n^{j_m})^2 \geq \rho_1^2 - \rho_m^2 + \varepsilon, \quad (12)$$

где $m = 1, 2, \dots, M$;

*) Соотношение (10) получается путем варьирования соотношения (2).

$$\min_{i', j'} \sum_{n=1}^N (g_n + \delta g_n) (x_n^{i'} - y_n^{j'})^2 \geq \min_{i, j} \sum_{n=1}^N g_n (x_n^i - y_n^j)^2. \quad (13)$$

Очевидно, что при выполнении соотношений (10) и (11) весовые коэффициенты

$$g'_n = g_n + \delta g_n \quad (n=1, 2, \dots, N) \quad (14)$$

являются неотрицательными и удовлетворяют условию нормирования (2).

Условие (12) означает, что величина ρ_1^2 , равная квадрату расстояния между множествами X и Y , увеличивается на величину ε^*). Однако при этом величины $\rho_2^2, \rho_3^2, \dots, \rho_M^2$ могут уменьшиться, так как правые части неравенств (12) могут принимать отрицательные значения. Соотношение (13) означает, что расстояние между множествами X и Y при найденных значениях весовых коэффициентов g'_1, g'_2, \dots, g'_N не уменьшается.

Из соотношения (10) найдем $\delta g_N = -\delta g_1 - \delta g_2 - \dots - \delta g_{N-1}$. Подставляя это выражение в (12), получим неравенства

$$\sum_{n=1}^{N-1} \delta g_n [(x_n^{i_m} - y_n^{j_m})^2 - (x_N^{i_m} - y_N^{j_m})^2] \geq \rho_1^2 - \rho_m^2 + \varepsilon,$$

которые в векторной форме можно записать в виде

$$\delta g A_m \geq b_m \quad (m=1, 2, \dots, M). \quad (15)$$

Здесь $\delta g (\delta g_1, \delta g_2, \dots, \delta g_{N-1})$ — неизвестный вектор, $A_m (a_{m1}, a_{m2}, \dots, a_{mN-1})$ — вектор с компонентами $a_{mn} = (x_n^{i_m} - y_n^{j_m})^2 - (x_N^{i_m} - y_N^{j_m})^2$, величина $b_m = \rho_1^2 - \rho_m^2 + \varepsilon$.

Отыскание вектора δg , при котором выполняются неравенства (15), производится с помощью модифицированного метода последовательных приближений, описанного в работе [5]. Пусть δg_1 — начальное значение вектора δg . Будем проверять последовательно при $m=M$,

*) Величина ε первоначально выбирается равной $\varepsilon = 2^{-4}$. Если при этом ε и принятых значениях $\delta g_1, \delta g_2, \dots, \delta g_N$ неравенства (12) не удовлетворяются, то величина ε полагается равной нулю.

$M-1, \dots, 1$ неравенства (15). Если при некотором m неравенство удовлетворяется, то вектор δg не меняется. В случае невыполнения m -го неравенства осуществляется исправление вектора δg по формуле

$$\delta g' = \delta g + c_m A_m, \quad (16)$$

где постоянная c_m выбирается из условия $(\delta g + c_m A_m) A_m \geq b_m$.

Полагая $c_m = [\lambda(b_m - \delta g A_m)] / A_m^2$, где $1 < \lambda < 2$, получим по формуле (16) новый вектор $\delta g'$, при котором m -е неравенство уже будет выполняться. Проверка следующих неравенств производится с вектором $\delta g = \delta g'$. В качестве приращений весовых коэффициентов берутся величины

$$\delta g_n = \alpha + g'_n \left[\sum_{n=1}^N \delta g'^2_n \right]^{-1/2}, \quad (17)$$

где $0 < |\alpha| \leq 1$. Здесь величины $\delta g'_n$ ($n = 1, 2, \dots, N$) получены с помощью решения системы неравенств (15);

величина $\delta g'_N = - \sum_{n=1}^{N-1} \delta g'_n$.

Если при вычислении значений весовых коэффициентов по формуле (14) оказалось, что $g_n + \delta g_n < 0$, то величина δg_n на данном шаге принимается равной $\delta g_n = -g_n$. Для этого величина α , входящая в формулу (17), полагается равной

$$\alpha = -g_n (\delta g'_n)^{-1} \left[\sum_{n=1}^N \delta g'^2_n \right]^{1/2} \quad (18)$$

и производится пересчет всех значений δg_n и g_n по формулам (17) и (14) соответственно. При этом если для некоторого n выполняется неравенство $|g'_n| < \varepsilon$ (ε — некоторая малая положительная величина), то на всех последующих циклах отыскания весовых коэффициентов величина δg_n полагается равной нулю, так что n -й признак в дальнейших вычислениях не учитывается. В этом случае для выполнения условия (2) осуществляется нормирование весовых коэффициентов по формуле

$$g_n = g'_n N [\sum g'_n]^{-1},$$

где суммирование производится по всем n , для которых $g'_n \geq \varepsilon$.

Процесс вычисления весовых коэффициентов на данном шаге заканчивается, если расстояние между множествами X и Y , вычисленное при этих значениях, не уменьшилось, т. е. если выполнилось неравенство (13). В противном случае процесс повторяется, причем в качестве начального вектора для проверки выполнения неравенств (15) берется вектор $\delta g'$. Как только неравенство (13) выполнится, вновь выбираются M пар точек из множеств X и Y , и весь цикл описанных операций повторяется.

Процесс вычисления коэффициентов g_1, g_2, \dots, g_N заканчивается, если расстояние между множествами X и Y на двух последовательных шагах увеличилось незначительно. При этом, как правило, имеется несколько пар точек из множеств X и Y , расстояния между которыми примерно одинаковы.

Общая блок-схема программы, реализующей описанный алгоритм, приведена на рис. 3; она состоит из нескольких блоков.

Блок *Б-1* предназначен для отыскания таких M пар точек x^{im} и y^{im} множеств X и Y , которые наиболее близко расположены друг к другу и для которых

$$\rho_1^2(x^{i_1}, y^{j_1}) \leq \rho_2^2(x^{i_2}, y^{j_2}) \leq \dots \leq \rho_M^2(x^{i_M}, y^{j_M}).$$

Вычисление величины $\rho(x, y)$ осуществляется по формуле (1), при этом учитываются только такие компоненты $(x_n - y_n)^2$, для которых соответствующие весовые коэффициенты g_n удовлетворяют условию $g_n \geq 2^{-10}$ (в первом цикле $g_1 = g_2 = \dots = g_N = 1$). В этом блоке вычисляется также расстояние между множествами X и Y , равное $\rho_*^2 = \rho_1^2(x^{i_1}, y^{j_1})$.

В блоке *Б-2* осуществляется подготовка данных для решения системы неравенств (15). Формируется матрица $\|a_{mn}\|$ ($m=1, 2, \dots, \dots, M, n=1, 2, \dots, N$), строки которой содержат компоненты векторов A_1, A_2, \dots, A_M , и вычисляются квадраты модулей этих векторов. Поскольку в каждой ячейке, содержащей информацию о больном, записано 14—15 кодов признаков (см. § 1), то получение величин a_{mn} связано с преобразованием кодов признаков в обычные двоичные числа для машины М-220. Эти величины a_{mn} находятся один раз и используются затем многократно при решении системы (15).

Блок *Б-3* предназначен для отыскания вектора приращений $\delta g(\delta g_1, \delta g_2, \dots, \delta g_N)$. В первой части блока производится поиск вектора δg , удовлетворяющего системе неравенств (15), во второй — нормирование этого вектора согласно формуле (17), где первоначально принято $\alpha=1$. При этом компоненты вектора δg , для которых весовые коэффициенты g_n удовлетворяют неравенству $|g_n| < 2^{-10}$, полагаются равными нулю. Проверка выполнения неравенств (15) производится последовательно: сначала проверяется M -е неравенство, затем $(M-1)$ -е неравенство и т. д. Каждый раз при не-

выполнении m -го неравенства осуществляется исправление вектора δg по формуле (16) (за исключением тех компонент δg_n , для которых $|g_n| < 2^{-10}$). Процесс заканчивается, если на данном шаге не произошло ни одного исправления вектора δg , либо если все M неравенств были проверены N раз. В последнем случае поиск вектора δg будет продолжен, если расстояние между множествами X и Y , вычисленное с учетом новых весовых коэффициентов, уменьшилось.

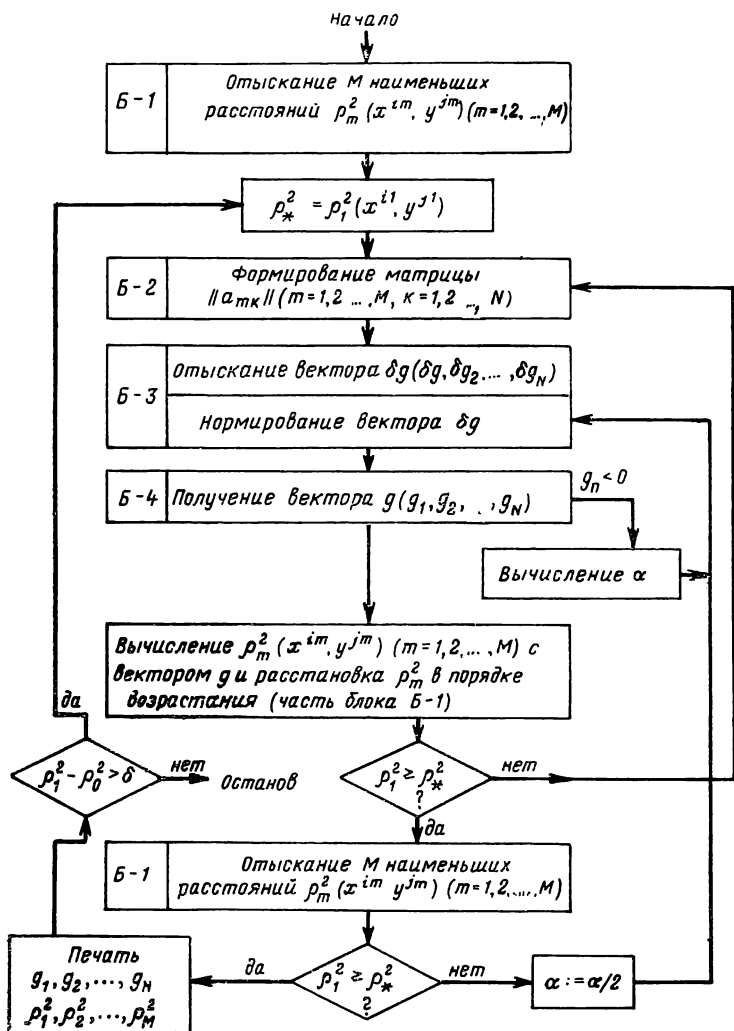


Рис. 3.

В блоке Б-4 производится вычисление и нормирование вектора $g(g_1, g_2, \dots, g_N)$ по формулам (14) и (19). Если при этом окажется выполненным неравенство $g_n + \delta g_n < 0$, то вычисляется величина α согласно выражению (18) и осуществляется переход к нормирующей части блока Б-3.

Затем, пользуясь новыми весовыми коэффициентами g_1, g_2, \dots, g_N , производится вычисление квадратов расстояний $\rho_m^2(x^{im}, y^{jm})$ между точками x^{im} и y^{jm} , отобранными в блоке Б-1, и расстояние их в порядке возрастания (используется часть программы, реализующей блок Б-1). Если $\rho_1^2 < \rho_*^2$, то осуществляется переход к блоку Б-2, где формируется матрица $\|a_{mn}\|$ в соответствии с новым порядком расстановки величин $\rho_m^2(x^{im}, y^{jm})$. При $\rho_1^2 \geq \rho_*^2$ находятся M наименьших расстояний $\rho_m^2(x^{im}, y^{jm})$ между точками x^{im} и y^{jm} множеств X и Y с новыми весовыми коэффициентами (используется блок Б-1). При $\rho_1^2 \geq \rho_*^2$ отыскание вектора $g_1(g_1, g_2, \dots, g_N)$ на данном цикле заканчивается. В противном случае величина α , входящая в равенство (17), уменьшается вдвое и осуществляется переход к нормирующей части блока Б-3.

На каждом цикле работы программы на печать выдаются значения весовых коэффициентов g_1, g_2, \dots, g_N и квадраты расстояний между точками множеств X и Y , наиболее близко расположенными друг к другу, причем $\rho_1^2 = \rho^2(X, Y)$.

Описанная программа реализована на машине БЭСМ-3М в двух вариантах. В первом варианте ($N \leq 40, M \leq 50$) она занимает 240 ячеек. Второй вариант программы рассчитан на случай, когда число признаков $N > 40$.

При этом коэффициенты g_1, g_2, \dots, g_N делятся на две группы $(g_1, g_2, \dots, g_{N_1})$ и $(g_{N_1+1}, g_{N_1+2}, \dots, g_N)$. К первой группе отнесены переменные весовые коэффициенты, которые находятся с помощью описанного алгоритма. Вторая группа содержит коэффициенты, не меняющиеся в процессе вычислений. При этом расстояния между точками x и y вычисляются по формуле

$$\begin{aligned} \rho^2(x, y) = \rho_1^2(x, y) + \rho_0^2(x, y) = \sum_{n=1}^{N_1} g_n (x_n - y_n)^2 + \\ + \sum_{n=N_1+1}^N g_n (x_n - y_n)^2. \end{aligned}$$

Поскольку величина ρ_0^2 постоянна, то для различных точек x^i и y^j она находится один раз перед началом работы программы. Значения $\rho_0^2(x^i, y^j)$ ($i = 1, 2, \dots, J_x, j = 1, 2, \dots, J_y$) записываются на барабан (по 5 чисел в каждую ячейку) и затем используются там, где это необходимо.

Процесс построения раздвигającej метрики осуществляется в несколько этапов. Сначала находится возможное раздвижение множеств

X и Y с коэффициентами g_1, g_2, \dots, g_{N_1} ($N_1 \leq 40$), затем с коэффициентами $g_{N_1+1}, g_{N_1+2}, \dots, g_{N_2}$ ($N_2 - N_1 \leq 40$) и т. д. Поскольку каждый раз, как правило, уменьшается размерность пространства признаков, то на последнем этапе учитываются лишь признаки, весовые коэффициенты которых $g_n \geq 2^{-10}$.

Второй вариант программы занимает 300 ячеек. В обоих случаях для записи статистического материала отведено 1300 ячеек МОЗУ. Для использования программы задаются следующие данные: J_x и J_y — число точек в множествах X и Y (при использовании второго варианта предполагается $J_x \times J_y \leq 5 \times 16000$), N и N_1 (для второго варианта) — количество признаков, M — число ближайших пар точек, x^{im} и y^{jm} , K — число ячеек, необходимое для записи кода одной точки, A_x и A_y — начальные адреса последовательностей $\{x^i\}$ и $\{y^j\}$.

ПОСТРОЕНИЕ РАЗДВИГАЮЩЕЙ НЕДИАГОНАЛЬНОЙ МЕТРИКИ

После того как из всей совокупности исходных признаков выделены признаки, наиболее важные для распознавания, естественно возникает задача отыскания таких линейных комбинаций этих признаков, которые наиболее существенны при классификации векторов. Обычно число существенных линейных комбинаций меньше общего числа исходных признаков. Выбирая в качестве новых признаков полученные существенные комбинации, можно тем самым сократить количество признаков, характеризующих векторы.

В основу метода построения раздвигающей недиагональной метрики положены те же соображения, что и при построении диагональной метрики. Распространение их на этот случай не представит особых затруднений, если квадратичную форму (7) после каждой операции изменения коэффициентов g_{mn} приводить к каноническому виду

$$\rho^2(x', y') = \sum_{n=1}^N g_n (x'_n - y'_n)^2, \quad (19)$$

для которого условие (6) и условие неотрицательности квадратичной формы (7) означают выполнение соотношений

$$g_n \geq 0, \quad g_1 + g_2 + \dots + g_N = N,$$

аналогичных соотношениям (11) и (12) в случае построения диагональной метрики. Для всякой квадратичной формы вида (7) приведение к каноническому виду

осуществляется при помощи некоторого невырожденного линейного преобразования переменных:

$$x'_n - y'_n = t_{n1}(x_1 - y_1) + t_{n2}(x_2 - y_2) + \dots + t_{nN}(x_N - y_N). \quad (20)$$

Отсюда ясно, что если некоторая величина g_n , входящая в равенство (19), мало отличается от нуля, то соответствующая линейная комбинация исходных признаков

$$t_{n1}(x_1 - y_1) + t_{n2}(x_2 - y_2) + \dots + t_{nN}(x_N - y_N)$$

несущественна, и, наоборот, при больших значениях g_n она имеет важное значение при распознавании.

Вычисление величин g_1, g_2, \dots, g_N и невырожденного линейного преобразования переменных x_1, x_2, \dots, x_N и y_1, y_2, \dots, y_N , матрица которого $T = \|t_{mn}\|$ ($m, n = 1, 2, \dots, N$), осуществляется последовательно в несколько шагов. Каждый шаг состоит из ряда описываемых ниже операций, выполнение которых приводит к некоторому увеличению расстояния между множествами X и Y .

Пусть $g^p (g_1^p, g_2^p, \dots, g_N^p)$ — вектор весовых коэффициентов и T^p — матрица преобразования переменных x_1, x_2, \dots, x_N и y_1, y_2, \dots, y_N в переменные x'_1, x'_2, \dots, x'_N и y'_1, y'_2, \dots, y'_N на p -м шаге. При $p=0$ в качестве компонент вектора g^0 используются весовые коэффициенты признаков, полученные при построении диагональной раздвигающей метрики, а в качестве матрицы T^0 — единичная квадратная матрица N -го порядка.

Из множеств X и Y выберем M пар точек x^{i_m} и y^{j_m} , наименее удаленных друг от друга, так чтобы

$$\rho_1^2(x^{i_1}, y^{j_1}) \leq \rho_2^2(x^{i_2}, y^{j_2}) \leq \dots \leq \rho_M^2(x^{i_M}, y^{j_M}).$$

Затем построим вектор $g^{p+1} (g_1^{p+1}, g_2^{p+1}, \dots, g_N^{p+1})$ и матрицу T^{p+1} , исходя из условий

$$\delta g_{11} + \delta g_{22} + \dots + \delta g_{NN} = 0; \quad (21)$$

$$g_n^{p+1} \geq 0 \quad (n = 1, 2, \dots, N); \quad (22)$$

$$\sum_{m=1}^N \sum_{n=1}^N \delta g_{mn} (x_m^{i_k} - y_m^{j_k}) (x_n^{i_k} - y_n^{j_k}) \geq \rho_1^2 - \rho_k^2 + \varepsilon, \quad (23)$$

где $k = 1, 2, \dots, M$;

$$\rho_{p+1}^2(X, Y) \geq \rho_p^2(X, Y). \quad (24)$$

Очевидно, что при выполнении этих соотношений квадратичная форма (19) и, следовательно, (7) на $(p+1)$ -м шаге останется неотрицательной, а расстояние между множествами X и Y не уменьшится.

Из условия (21) найдем величину $\delta g_{NN} = - \sum_{n=1}^{N-1} \delta g_{nn}$ и подставим ее в (23). Тогда это неравенство можно представить в векторной форме в виде

$$\delta g A_k \geq b_k, \quad (25)$$

где $\delta g (\delta g_{11}, \delta g_{12}, \dots, \delta g_{1N}, \delta g_{21}, \dots, \delta g_{N, N-1})$ — неизвестный вектор, $A_k (a_{11}, a_{12}, \dots, a_{1N}, a_{21}, a_{22}, \dots, a_{NN-1})$ — вектор с компонентами

$$a_{mn} = (x'_m{}^{i_k} - y'_m{}^{j_k})(x'_n{}^{i_k} - y'_n{}^{j_k}) \text{ при } n \neq m,$$

$$a_{nn} = (x'_n{}^{i_k} - y'_n{}^{j_k})^2 - (x'_N{}^{i_k} - y'_N{}^{j_k})^2 \text{ при } n = m,$$

а величина $b_k = \rho_1^2 - \rho_k^2 + \varepsilon$.

Для отыскания вектора δg применяется метод, описанный в предыдущем параграфе. Будем проверять выполнение неравенств (25) последовательно при $k=1, 2, \dots$ и каждый раз в случае его невыполнения осуществлять исправление компонент вектора δg по формулам

$$\delta g'_{mn} = \delta g_{mn} + c_m a_{mn}, \quad \delta g'_{NN} = - \sum_{n=1}^{N-1} \delta g'_{nn}, \quad (26)$$

где постоянная $c_k = \lambda A_k^{-2} (b_k - \delta g A_k)$, $1 < \lambda < 2$.

Образуем матрицу квадратичной формы вида (7):

$$G^{p+1} = \begin{vmatrix} g_1^p + \delta g_{11} & \delta g_{21} & \dots & \delta g_{1N} \\ \delta g_{21} & g_2^p + \delta g_{22} & \dots & \delta g_{2N} \\ \delta g_{N1} & \delta g_{N2} & \dots & g_N^p + \delta g_{NN} \end{vmatrix},$$

где величина δg_{mn} определяется равенством

$$\delta g_{mn} = \alpha \delta g'_{mn} |\delta g'|^{-1} \quad (0 < |\alpha| \leq 1). \quad (27)$$

Матрица G^{p+1} симметрическая, так как $\delta g_{mn} = \delta g_{nm}$ при всех $n \neq m$. Как известно [6], любая действительная

симметрическая матрица ортогонально подобна действительной диагональной матрице. Следовательно, для квадратичной формы с матрицей G^{p+1} можно найти ортогональное преобразование переменных x'_1, x'_2, \dots, x'_N и y'_1, y'_2, \dots, y'_N , приводящее ее к каноническому виду

$$\rho^2(x'', y'') = \sum_{n=1}^N g_n^{p+1} (x''_n - y''_n)^2, \quad (28)$$

где

$$x''_n - y''_n = l_{n1}(x'_1 - y'_1) + l_{n2}(x'_2 - y'_2) + \dots \\ \dots + l_{nN}(x'_N - y'_N). \quad (29)$$

При таком преобразовании весовые коэффициенты g_n^{p+1} ($n = 1, 2, \dots, N$) являются собственными значениями матрицы G^{p+1} , а соответствующие им собственные векторы $\{l_{n1}, l_{n2}, \dots, l_{nN}\}$ составляют ортонормированную систему векторов.

Присвоим коэффициентам t_{mn} в равенствах (20) индекс p и подставим полученные выражения в (29), тогда получим

$$x''_n - y''_n = \sum_{j=1}^N l_{nj} t_{j1}^p (x_1 - y_1) + \sum_{j=1}^N l_{nj} t_{j2}^p (x_2 - y_2) + \dots + \\ + \sum_{j=1}^N l_{nj} t_{jN}^p (x_N - y_N).$$

Отсюда следует, что матрица T^{p+1} равна произведению матрицы преобразования (29) и матрицы T^p :

$$T^{p+1} = \begin{vmatrix} t_{11}^{p+1} & t_{12}^{p+1} & \dots & t_{1N}^{p+1} \\ t_{21}^{p+1} & t_{22}^{p+1} & \dots & t_{2N}^{p+1} \\ \dots & \dots & \dots & \dots \\ t_{N1}^{p+1} & t_{N2}^{p+1} & \dots & t_{NN}^{p+1} \end{vmatrix} = \begin{vmatrix} l_{11} l_{12} \dots l_{1N} \\ l_{21} l_{22} \dots l_{2N} \\ \dots & \dots & \dots \\ l_{N1} l_{N2} \dots l_{NN} \end{vmatrix} \times \begin{vmatrix} t_{11}^p & t_{12}^p & \dots & t_{1N}^p \\ t_{21}^p & t_{22}^p & \dots & t_{2N}^p \\ \dots & \dots & \dots & \dots \\ t_{N1}^p & t_{N2}^p & \dots & t_{NN}^p \end{vmatrix}.$$

Если в равенстве (28) величина g_n^{p+1} окажется отрицательной, то значение α , входящее в выражение (27), полагается равным

$$\alpha = \frac{g_n^p}{g_n^p - g_n^{p-1}} \approx \frac{1}{2^s} \quad (s = 0, 1, 2, \dots) \quad (30)$$

и производится пересчет по формуле (27) всех значений δg_{mn} , а затем и коэффициентов $g_1^{p+1}, g_2^{p+1}, \dots, g_N^{p+1}$. При таком способе задания значения δg_{mn} с увеличением s уменьшаются и величина g_n^{p+1} становится неотрицательной. Если выполняется условие $|g_n^{p+1}| < \epsilon$ (ϵ — малое положительное число), то в дальнейших вычислениях весовой коэффициент g_n не учитывается, так что на $p+2, p+3, \dots$ шагах приращения δg_{mn} и δg_{nm} полагаются равными нулю. При этом матрица (29) преобразуется так, что система собственных векторов, составляющих ее строки, остается ортонормированной: элементы l_{mn} и $l_{nm} (m \neq n)$, стоящие в n -й строке и n -м столбце, полагаются равными нулю, а элемент l_{nn} , стоящий на их пересечении, равным единице.

Процесс отыскания вектора g^{p+1} и матрицы T^{p+1} считается законченным, если удовлетворяется неравенство (24). В противном случае производится повторное решение системы неравенств (25), причем в качестве начального значения вектора δg принимается вектор $\delta g'$. Поиск раздвигающей метрики описанным путем осуществляется до тех пор, пока не выполнится неравенство

$$\rho_{p+1}^2(x, Y) - \rho_p^2(X, Y) < \delta,$$

где δ — малое положительное число.

При проведении «экзамена» для точки y с помощью алгоритма по общей близости признаков заболевания (см. § 2) теперь вместо метрики вида (1) используется метрика вида (19), в которой величины $x'_n - y'_n$, согласно равенству (20), представляют собой линейные комбинации исходных признаков.

Общая блок-схема программы, реализующей алгоритм построения раздвигающей недиагональной метрики (рис. 4), аналогична блок-схеме программы, реализующей алгоритм построения диагональной метрики (см. рис. 3). Здесь расстояние между точками x и y находится по формуле (19), где разности $x'_n - y'_n$ определены равенствами (20). Поиск собственных значений g_1, g_2, \dots, g_N и собственных векторов $\|l_{mn}\|$ ($m, n = 1, 2, \dots, N$) осуществляется с помощью стандартной программы СП-135. Если окажется выполненным неравенство $g_n < 0$, то производится вычисление величины α согласно формуле (30) и вновь осуществляется нормирование век-

тора δg . При $|g_n| < 2^{-10}$ производится преобразование матрицы $\|l_{mn}\|$: элементы l_{mn} и $l_{nm} (n \neq m)$ полагаются равными нулю, а элемент l_{nn} — равным единице

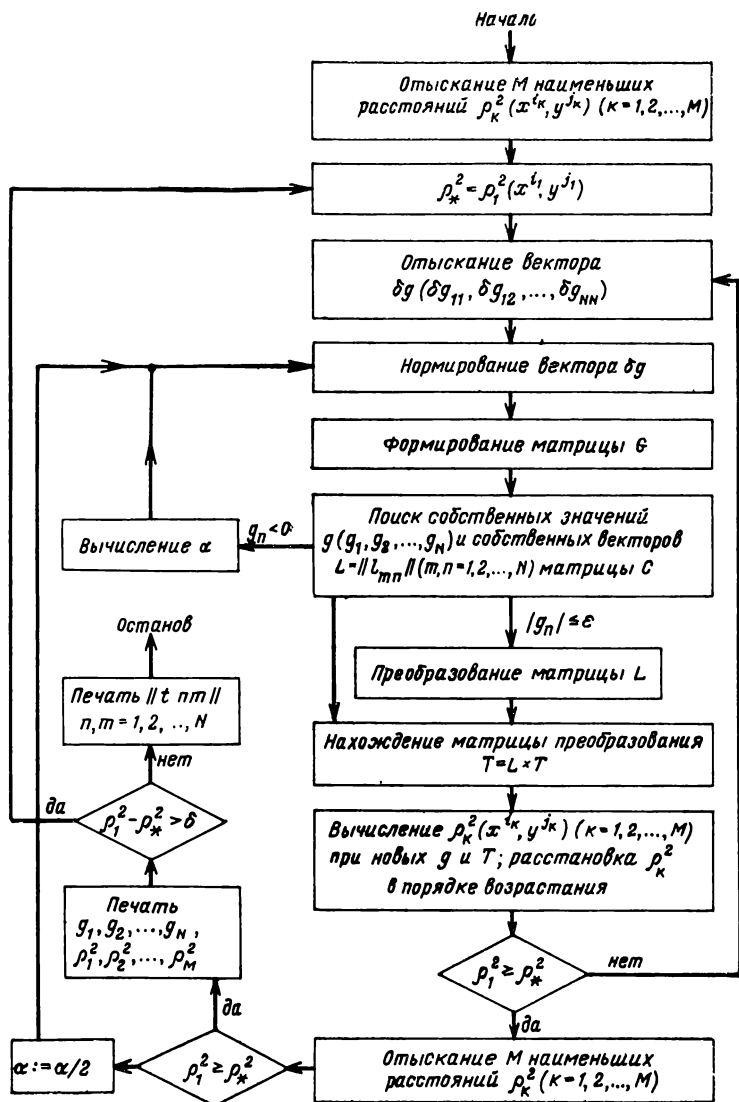


Рис. 4.

Программа реализована для случая $N \leq 24$, $M \leq 40$ и занимает вместе с формирующей частью и необходимыми для ее работы константами 340 ячеек. Для размещения статистического материала в МОЗУ отведено 800 ячеек. При использовании этой программы задаются те же данные, что и в случае построения диагональной раздвигающей метрики (см. § 3).

ПОИСК ПЕРЕМЕННЫХ ВЕСОВЫХ КОЭФФИЦИЕНТОВ

Описываемый ниже алгоритм отыскания весовых коэффициентов основан на предположении, что в каждой точке y пространства E_N значение параметра π может быть найдено по значению π_i в близкой к y точке x^i из обучающей последовательности

$$\{x^i(x_1^i, x_2^i, \dots, x_N^i)\} (i = 1, 2, \dots, J)$$

по формуле

$$\pi = \pi_i + (\text{grad } \pi_i \bullet x^i y). \quad (31)$$

Здесь под $\text{grad } \pi_i$ понимается вектор, компоненты которого (g_1, g_2, \dots, g_N) равны скорости изменения параметра π в точке x^i в направлении соответствующих координатных осей. Если некоторая компонента g_n градиента мала по абсолютной величине по сравнению с другими, то это означает, что признак x_n в меньшей степени влияет на значение параметра π и поэтому ему следует приписать меньший вес.

В зависимости от расположения точек в пространстве признаков скорости изменения параметра π в направлении координатных осей могут быть различными, поэтому и весовые коэффициенты в разных точках пространства признаков могут менять свое значение. Это означает, что для разных больных один и тот же признак заболевания может иметь различное влияние на диагноз или прогноз. Например, при прогнозировании состояния больного после резекции легкого в одних случаях наибольшее влияние на прогноз имеют возраст и пол больного, в других — жизненная емкость легких, дыхательная пауза на вдохе и выдохе [1, 3].

Алгоритм отыскания весовых коэффициентов g_1, g_2, \dots, g_N состоит из трех частей:

- нахождение опорных точек;
- поиск весовых коэффициентов в опорных точках;
- определение весовых коэффициентов в произвольной точке пространства E_N .

В первой и второй частях используется обучающая последовательность, третья — опирается на результаты решения первых двух.

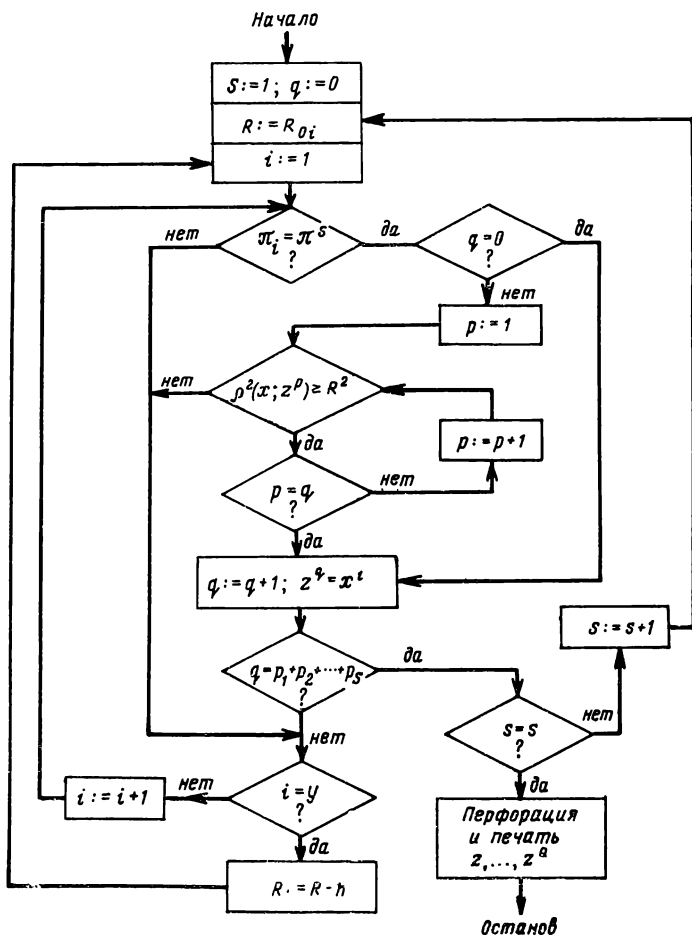


Рис. 5.

Нахождение опорных точек. Точки z^1, z^2, \dots, z^Q обучающей последовательности назовем опорными, если для некоторого значения R расстояния между любыми двумя точками больше R и если каждая точка x^i обучающей последовательности находится от одной из этих точек на расстоянии, не превышающем R .

Количество опорных точек в каждом s -м классе со значением параметра π^s ($s=1, 2, \dots, S$) определяется заданной величиной p_s , так что общее их число $Q = \sum_{s=1}^S p_s$.

Величина R находится в результате поиска точек $\{z^q\}$; первоначально она выбирается равной величине наибольшего расстояния между точками обучающей последовательности *).

Отыскание опорных точек производится в соответствии с блок-схемой, приведенной на рис. 5. Предполагается, что точки обучающей последовательности расположены случайно, поэтому для каждой точки x^i , начиная с x^1 , осуществляется проверка совпадения значения π_i со значением параметра π , который последовательно принимает заданные значения: $\pi^1, \pi^2, \dots, \pi^s$. В качестве первой опорной точки z^1 выбирается первая точка x^i , для которой $\pi_i = \pi^1$. Для каждой следующей точки x^i проверяется выполнение двух условий: $\pi_i = \pi^1$ и $\rho(z^1, x^i) \geq R$, где величина $\rho(x, y)$ вычисляется по формуле (1) при $g_n = 1$.

Точка x^i , для которой впервые выполняются оба условия, принимается за опорную точку z^2 . Для получения третьей опорной точки определяется точка x^i , для которой выполняются условия $\pi_i = \pi_1$ и $\rho(x^i, z^q) \geq R$ ($q=1, 2$) и т. д. Если обучающая последовательность x^1, x^2, \dots, x^J не содержит точек, удовлетворяющих этим условиям, то величина R уменьшается на некоторое число h и цикл повторяется. Таким образом находятся p_1 опорных точек первого класса.

Отыскание опорных точек второго класса производится при значении $\pi = \pi^2$. При этом для каждой точки x^i осуществляется проверка условий

$$\pi_i = \pi^2 \text{ и } \rho(z^q, x^i) \geq R \text{ (по всем } q \text{).}$$

*) Это требование не является обязательным, оно может быть заменено условием $R_0 = (\alpha_1^2 + \alpha_2^2 + \dots + \alpha_N^2)^{1/2}$, где α_n — наибольшее значение градации признака x_n .

Величина R принимается снова равной начальному значению R_0 . Аналогично находятся опорные точки остальных классов z^1, z^2, \dots, z^Q , которые затем выдаются на печать и устройство перфорации.

Программа вместе с формирующей частью и константами занимает 85 ячеек. Для ее использования должны быть заданы следующие данные: R_0 и h — начальное значение и шаг уменьшения величины R , N — количество признаков, K — количество ячеек для записи кода одной точки, J — длина обучающей последовательности, s — количество классов, π^s и p_s — значение параметра π и количество необходимых опорных точек s -го класса соответственно.

Поиск весовых значений признаков в опорных точках. Весовые коэффициенты g_1, g_2, \dots, g_N в опорных точках z^1, z^2, \dots, z^Q находятся методом наименьших квадратов, который позволяет сгладить влияние случайных ошибок в обучающей последовательности достаточной длины и получить наилучшее приближение к линейной зависимости.

Для каждой опорной точки z^q ($q = 1, 2, \dots, Q$) поиск весовых коэффициентов осуществляется согласно блок-

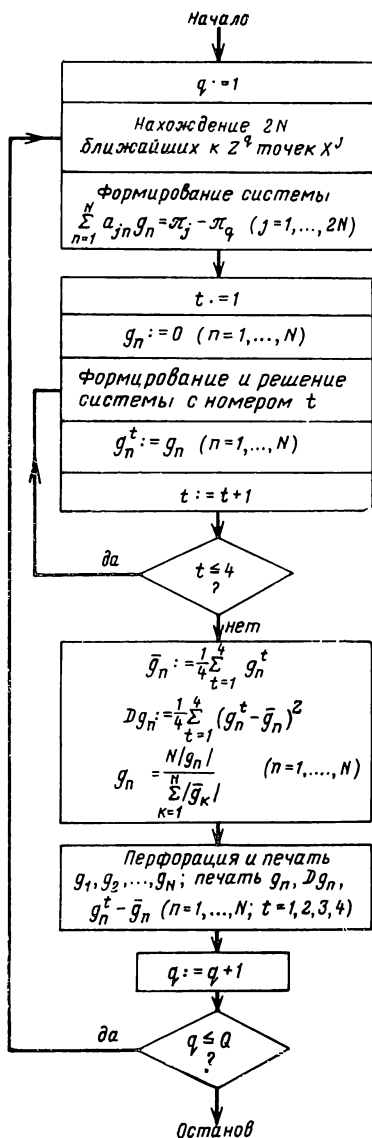


Рис. 6.

схеме, изображенной на рис. 6. Сначала для опорной точки $z^q (z_1^q, z_2^q, \dots, z_N^q)$ находится $2N$ ближайших точек $x^j (x_1^j, x_2^j, \dots, x_N^j)$ и в соответствии с формулой (31) формируется система $2N$ линейных уравнений с N неизвестными компонентами градиента g_1, g_2, \dots, g_N :

$$\sum_{n=1}^N a_{jn} g_n = \pi_j - \pi_q \quad (j=1, 2, \dots, 2N),$$

где $a_{jn} = z_n^j - z_n^q$. Затем из этих уравнений составляются четыре подсистемы, каждая из которых состоит из $P = [3N/2]$ уравнений и отличается от остальных не менее чем на $[N/2]$ уравнения. Решение каждой подсистемы находится методом наименьших квадратов:

$$u(g_1, g_2, \dots, g_N) = \sum_{j=1}^P (a_{j1}g_1 + a_{j2}g_2 + \dots + a_{jN}g_N - \pi_j + \pi_q)^2 = \min.$$

Минимум функции $u(g_1, g_2, \dots, g_N)$ определяется методом последовательных приближений по формуле

$$g_n^{v+1} = g_n^v - \lambda_v \frac{\partial u(g_1^v, g_2^v, \dots, g_N^v)}{\partial g_n^v} \quad (n=1, 2, \dots, N; v=0, 1, 2, \dots).$$

Здесь

$$\frac{\partial u}{\partial g_n^v} = 2 \sum_{j=1}^P (a_{j1}g_1^v + a_{j2}g_2^v + \dots + a_{jN}g_N^v - \pi_j + \pi_q) a_{jn},$$

$$\lambda_v = \frac{u(g_1^v, g_2^v, \dots, g_N^v)}{\sum_{n=1}^N \left[\frac{\partial u(g_1^v, g_2^v, \dots, g_N^v)}{\partial g_n^v} \right]^2}.$$

Начальные значения неизвестных весовых коэффициентов g_1, g_2, \dots, g_N для первой подсистемы задаются нулевыми, а для каждой из остальных трех подсистем в качестве начальных значений принимаются решения предыдущих подсистем. Последовательное уточнение решений каждой из четырех подсистем проводится до тех пор, пока не будут выполнены условия

$$|g_n^{v+1} - g_n^v| < 0,1 |g_n^{v+1}| \quad (n=1, 2, \dots, N).$$

В качестве весовых коэффициентов признаков для опорной точки z^q выбираются величины

$$g_n = N |\bar{g}_n| \left[\sum_{m=1}^N \bar{g}_m \right]^{-1},$$

где \bar{g}_n — среднее арифметическое решений g_n^t ($t = 1, 2, 3, 4$) по всем четырем подсистемам.

Для каждой опорной точки, кроме весовых коэффициентов признаков, печатаются следующие данные (по всем четырем подсистемам):

- 1) g_n ($n = 1, 2, \dots, N$) — среднее арифметическое коэффициентов g_n ;
- 2) Dg_n — дисперсии решений;
- 3) отклонения от среднего $\bar{g}_n - g_n$.

Величины дисперсий и отклонений от средних позволяют оценить степень приближения линейной зависимости признаков к параметру π .

Программа реализована для случая $Q \times K \leq 64$, $N \leq 32$, $J \times K \leq 896$; она занимает 280 ячеек.

Для использования этой программы должны быть заданы опорные точки z^1, z^2, \dots, z^Q , J — количество обучающих точек x^i , N — количество признаков и K — количество ячеек для записи кода одной точки.

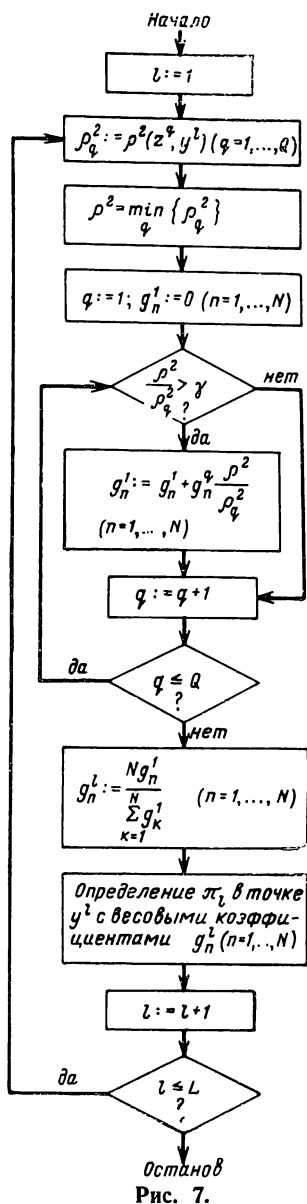
Нахождение весовых коэффициентов признаков в произвольной точке. В точке y пространства E_N весовые коэффициенты признаков принимаются равными весовым коэффициентам в ближайшей опорной точке. Если точка y близка к нескольким опорным точкам, то в качестве g_1, g_2, \dots, g_N используются линейные комбинации весовых коэффициентов в этих точках, причем чем ближе точка y к опорной точке, тем больше ее весовые коэффициенты.

Алгоритм нахождения весовых коэффициентов g_1, g_2, \dots, g_N в точке y состоит в вычислении их по формуле

$$g_n = N g'_n \left[\sum_{m=1}^N g'_m \right]^{-1},$$

где

$$g'_n = \sum_{q=1}^Q a_q g_n^q, \quad a_q \begin{cases} 0, & \text{если } \rho^2/\rho_q^2 \leq \gamma; \\ \rho^2/\rho_q^2, & \text{если } \rho^2/\rho_q^2 > \gamma; \end{cases}$$



$\rho_q = \rho(z^q, y)$, $\rho = \min \rho_q$, $0 < \gamma < 1$, $g_1^q, g_2^q, \dots, g_n^q$ — весовые коэффициенты признаков в точке z^q .

Для удобства использования программа, реализующая эту часть алгоритма, совмещена с программой определения значения параметра π в экзаменационной последовательности y^l ($l=1, 2, \dots, L$) с помощью алгоритма диагностики и прогнозирования по общей близости признаков (см. § 2).

Блок-схема третьей части алгоритма приведена на рис. 7.

Для работы программы должны быть заданы: J — число обучающих точек, L — число экзаменационных точек, Q — количество опорных точек и их координаты, N — число признаков, K — количество ячеек для записи кода одной точки, γ — параметр алгоритма. Предполагается, что

$$JK \leq 1345,$$

$$L \leq 100,$$

$$QK \leq 64,$$

$$QN \leq 672.$$

СПИСОК ЛИТЕРАТУРЫ

1. Неймарк Ю. И., Баталова З. С. Опыт использования быстродействующей вычислительной машины для медицинской диагностики, прогнозирования исхода оперативного вмешательства или заболевания и выбора оптимального метода лечения. Ученые записки НИИ ПМК и ф-та ВМиК Горьковского Государственного университета, 1967.
2. Нильсон Н. Обучающиеся машины. Изд-во «Мир», 1967.
3. Курош А. Г. Курс высшей алгебры. Физматгиз, 1963.

А. А. Дорофеюк, В. Я. Лумельский

Реализация алгоритмов обучения распознаванию образов «без учителя» на ЭВМ

Во многих практических задачах возникает необходимость проводить разделение (классификацию) различных образов. Например, в метеорологии такими образами могут быть параметры, характеризующие состояние атмосферы, в минералогии — химические анализы минерала, в геофизике — геофизические данные каротажа скважин, в медицине — параметры, характеризующие состояние больного, и т. д.

Подобную классификацию часто проводят в тех случаях, когда есть некоторые предварительные соображения о существовании нескольких классов подобных образов (различные типы погоды, группы минералов, режимы работы технических агрегатов, различные болезни и т. д.).

Пусть каждый объект x характеризуется набором r признаков (x^1, \dots, x^r) , т. е. является r -мерным вектором. Введем пространство векторов X , в котором каждому вектору x соответствует точка с координатами (x^1, \dots, \dots, x^r) . Часто оказывается, что различным классам образов соответствуют изолированные, достаточно удаленные друг от друга группы векторов в пространстве X .

В задачах подобного типа можно предложить следующий метод классификации: образы, которым соответствуют векторы из одной такой группы, относятся к одному и тому же классу. Было предложено несколько алгоритмов, позволяющих выделять в пространстве X достаточно компактные группы векторов. Ниже приводится описание программ, реализующих некоторые из этих алгоритмов [1].

1. ОПИСАНИЕ АЛГОРИТМОВ

Алгоритм «Объединение»

При описании алгоритмов без ограничения общности можно предполагать, что требуется произвести разделение векторов на два класса: A и B .

В алгоритме применяется потенциальная функция $K(x, y)$, зависящая от расстояния $R(x, y)$ между векторами x и y в пространстве X . В данном случае $K(x, y)$ убывает с ростом расстояния $R(x, y)$ [2].

Используя эту функцию, введем понятие *меры близости* $K(x, D)$ вектора x к конечному множеству векторов D :

$$K(x, D) = \frac{1}{N_D} \sum_{x_j \in D} K(x, x_j), \quad (1)$$

где N_D — число векторов $x_j \in D$.

Введенная мера близости между вектором x_j и множеством D отвечает интуитивному понятию «близость», которым мы пользуемся при классификации.

Аналогичным образом введем понятие *меры близости* $K(C, D)$ между двумя множествами точек C и D :

$$K(C, D) = \frac{1}{N_C N_D} \sum_{x_i \in C} \sum_{x_j \in D} K(x_i, x_j), \quad (2)$$

где N_C — число векторов $x_i \in C$; N_D — число векторов $x_j \in D$.

Предположим, что на вход машины поступило N векторов x_1, \dots, x_N , которые необходимо разделить на K классов. Алгоритм определяется индуктивно:

1. Пусть к l -му шагу алгоритма векторы x_1, \dots, x_N разделены на K_l непересекающихся множеств A_1, \dots, A_{K_l} .

На этом шаге производится разделение векторов на $(K_l - 1)$ множеств, для чего объединяются два множества A_i и A_j , для которых мера близости $K(A_i, A_j)$ максимальна:

$$K(A_i, A_j) = \max_{p, q \neq p} K(A_p, A_q), \quad p, q = 1, \dots, K_l.$$

Полученные множества перенумеровываются: A_1, \dots, A_{K_l-1} .

2. При $K_l = K$ работа алгоритма прекращается.

3. В качестве начального разделения $A_1, \dots, A_{K_{нач}}$

можно принять разделение, при котором $K_{\text{нач}} = N$, т. е. каждый класс A_i состоит из единственного вектора входной последовательности.

Однако для больших N такое начальное разделение значительно увеличивает время работы алгоритма. Сокращение этого времени можно осуществить с помощью алгоритма, который производит предварительную классификацию векторов на $K_{\text{нач}}$ классов.

Алгоритм «Спектр»

Работа алгоритма происходит в два этапа:

На первом этапе множеству векторов $M_0 = \{x_1, \dots, x_N\}$, поступивших на вход машины, ставятся в соответствие две последовательности $K_N = \{\tilde{x}_1, \dots, \tilde{x}_n\}$ и $Q_N = \{\tilde{K}_1, \dots, \tilde{K}_N\}$ по следующему индуктивному правилу:

1. В начале $\tilde{x}_1 = x_1$, $\tilde{K}_1 = 1$. Затем вектор x_1 исключается из множества M_0 . Полученное множество обозначается M_1 .

2. Пусть к $(m+1)$ -му шагу алгоритма из исходного множества M_0 отобрано m векторов и построены последовательности $K_m = \{\tilde{x}_1, \dots, \tilde{x}_m\}$ и $Q_m = \{\tilde{K}_1, \dots, \tilde{K}_m\}$, а оставшиеся векторы входной последовательности объединены во множество M_m . Тогда на $(m+1)$ -м шаге строятся последовательности

$$\mu_{m+1} = \{\tilde{x}_1, \dots, \tilde{x}_m, \tilde{x}_{m+1}\} \text{ и}$$

$$Q_{m+1} = \{\tilde{K}_1, \dots, \tilde{K}_m, \tilde{K}_{m+1}\}.$$

Здесь

$$\tilde{x}_{m+1} = x_j \left| \max_{x_j \in M_0/M_m} (\mu_m, x_j), \tilde{K}_{m+1} = K(\mu_m, \tilde{x}_{m+1}). \quad (3)$$

Другими словами, работа алгоритма заключается в следующем: вначале выбирается вектор x_1 , затем среди оставшихся $(N-1)$ векторов определяется вектор x_2 , ближайший к x_1 , далее среди оставшихся $(N-2)$ векторов находится вектор x_3 , ближайший в смысле (1) к первым двум (т. е. ближайший к множеству $\tilde{x}_1 \cup \tilde{x}_2$) и т. д.

При этом каждому вектору \tilde{x}_j ставится в соответствие величина \tilde{K}_j из последовательности Q .

Как видно из выражения (3), \tilde{K}_j является мерой близости выбираемого на j -м шаге вектора \tilde{x}_j к множеству всех ранее отобранных векторов $\{\tilde{x}_1, \dots, \tilde{x}_{j-1}\}$.

Такое упорядочение обладает следующей важной

особенностью. Допустим, что последовательность $\{x_1, \dots, \dots, x_N\}$ является набором векторов двух достаточно удаленных друг от друга множеств A и B . В этом случае, если, например, $x_1 \in A$, то, как правило, сначала будут отобраны все векторы, принадлежащие множеству A , а затем — векторы, принадлежащие множеству B . Причем на границе перехода между A и B , т. е. в тот момент, когда будет впервые отобран вектор $\tilde{x}_j \in B$, величина \tilde{K}_j скачкообразно уменьшается.

Эти соображения относительно членов последовательности Q положены в основу приводимого ниже второго этапа работы алгоритма начального разделения.

На втором этапе происходит разделение последовательности на множества (ядра) $A_1, \dots, A_{K_{\text{нач}}}$. Для этого последовательность величин \tilde{K}_j преобразуется в последовательность

$$\Delta_j = \frac{(\tilde{K}_j - \tilde{K}_{j+1})}{\tilde{K}_j}, \quad j = 2, \dots, N - 1. \quad (4)$$

Затем выбирается граничное значение $\Delta_{\text{гр}}$, по которому, используя последовательность Δ_j , производится разделение множества $\{\tilde{x}_1, \dots, \tilde{x}_N\}$ на $K_{\text{нач}}$ ядер.

Для работы алгоритма «Спектр» необходимо заранее задать число классов, на которое производится разделение векторов. Однако существует способ (описанный ниже), с помощью которого это число выбирается машиной в процессе обработки входной последовательности.

В дальнейшем используется величина $K(D, D)$, характеризующая среднюю меру близости между векторами x_j из конечного множества D :

$$K(D, D) = \frac{2}{N_D(N_D - 1)} \sum_{i=1}^{N_D-1} \sum_{j>i} K(x_i, x_j), \quad (5)$$

где N_D — число точек $x_j \in D$.

Величина $K(D, D)$ тем больше, чем «компактнее» множество D . Очевидно, что деление (классификация) тем лучше, чем ближе друг к другу расположены векторы внутри каждого класса.

В качестве средней (по всем классам) меры близости между векторами внутри класса можно принять, например, величину

$$J_1 = \frac{1}{N} \sum_{i=1}^S n_{A_i} K(A_i, A_i), \quad (6)$$

где N — общее число векторов; n_{A_i} — число векторов в классе A ; S — число классов.

Таким образом, разделение (классификация) тем лучше, чем больше (при прочих равных условиях) величина J_1 .

Кроме того, ясно, что классификация тем лучше, чем дальше расположены друг от друга классы, получившиеся в результате разделения.

Примем за *среднюю меру близости между классами* величину

$$J_2 = \frac{1}{n_1} \sum_{i=1}^S \sum_{j>i}^S n_{A_i} n_{A_j} K(A_i, A_j), \quad (7)$$

где $n_1 = \sum_{i=1}^S \sum_{j>i}^S n_{A_i} n_{A_j}$.

То есть разделение тем лучше, чем меньше (при прочих равных условиях) величина J_2 .

Очевидно, что критерий оценки качества полученной классификации должен учитывать обе части величины, т. е. должен иметь вид

$$J = f(J_1, J_2). \quad (8)$$

Причем необходимо, чтобы величина J увеличивалась с увеличением J_1 (при фиксированном J_2) и уменьшалась с увеличением J_2 (при фиксированном J_1).

Пусть на вход машины поступила последовательность векторов x_1, \dots, x_N . Используя алгоритмы «Спектр» и «Объединение», последовательно классифицируем эти векторы на $K, K-1, \dots, 2$ классов. Лучшей среди этих $(K-1)$ классификаций считается та, для которой величина J максимальна.

2. ОПИСАНИЕ ПРОГРАММ, РЕАЛИЗУЮЩИХ АЛГОРИТМЫ «СПЕКТР» И «ОБЪЕДИНЕНИЕ»

Последовательность работы программ ясна из блок-схем (рис. 1, 2). (На рис. 1 приведена блок-схема программы, реализующей алгоритм «Спектр». На

рис. 2 изображена блок-схема программы получения начального разделения и программы, реализующей алгоритм «Объединение».) На блок-схемах приняты сле-

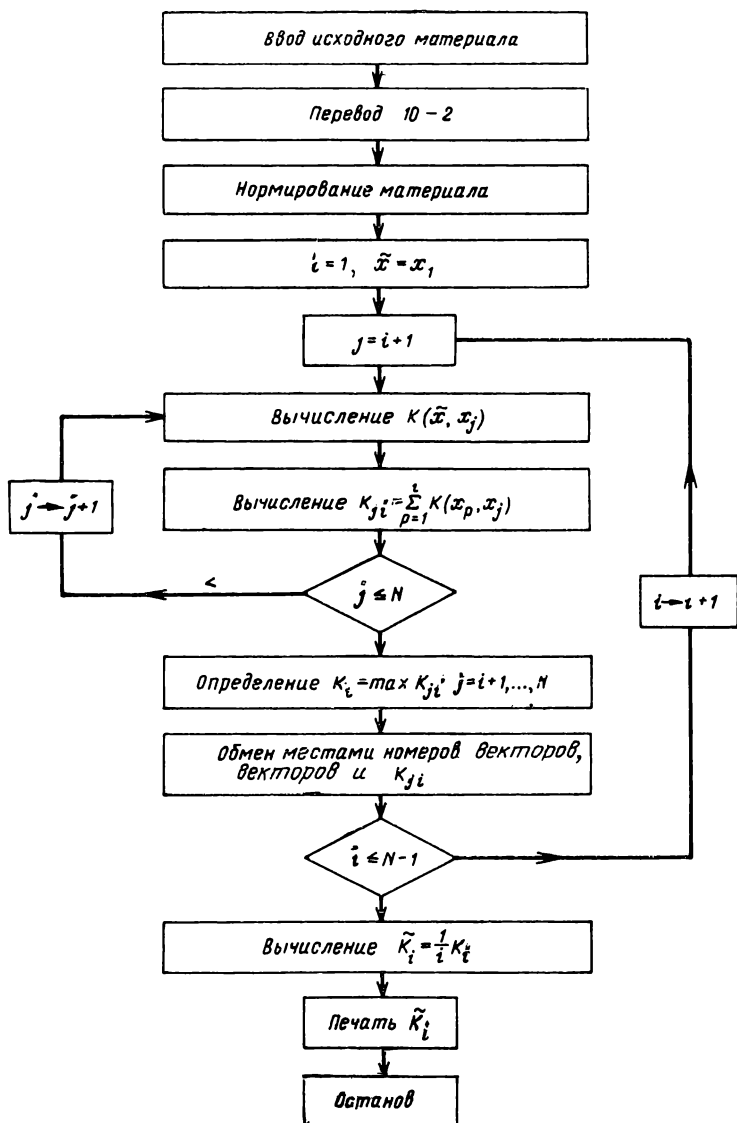


Рис. 1.

дующие обозначения: N — число векторов; a — целое положительное число, определяющее выбор $\Delta_{\text{гр}}$, b — число ядер A_j , содержащих по одному вектору; S — общее

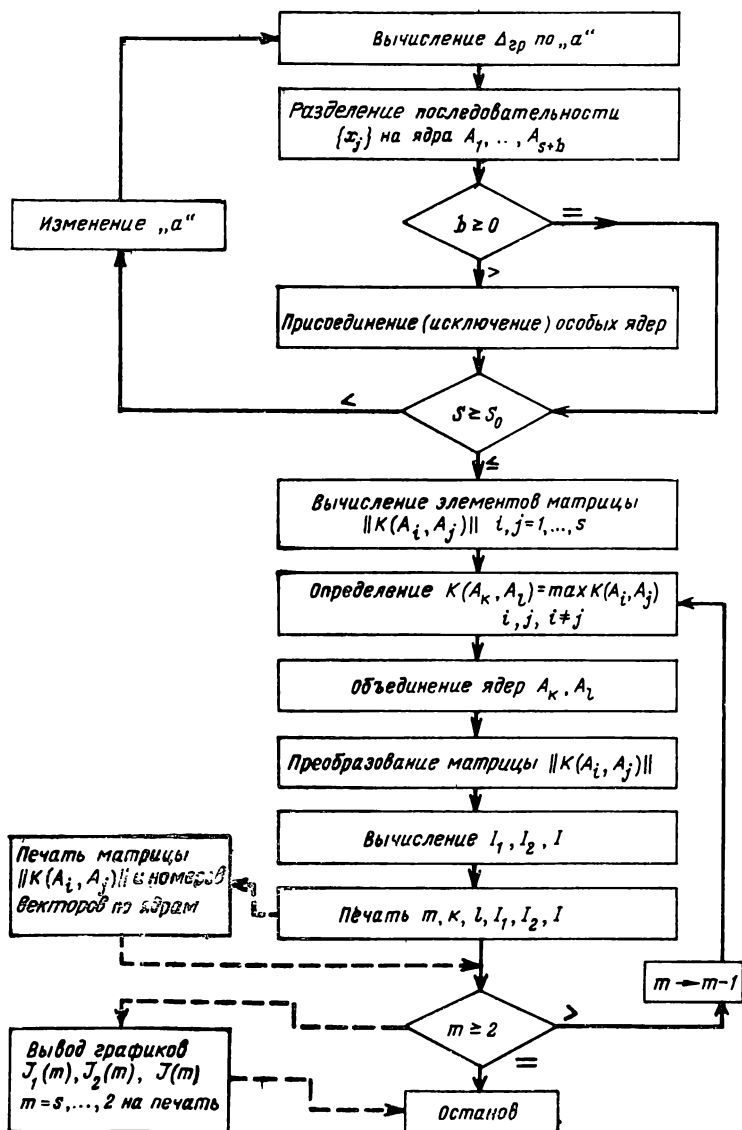


Рис. 2.

число ядер; S_0 — максимально допустимое число ядер;
 m — текущий порядок матрицы $\|K(A_i, A_j)\|$.

3. НЕКОТОРЫЕ РЕКОМЕНДАЦИИ ПО ПРОГРАММНОЙ РЕАЛИЗАЦИИ АЛГОРИТМОВ

Кодирование и размещение входной последовательности

Входную последовательность векторов целесообразно размещать в памяти машины одним из двух способов:

$$1) \ x_1^1, x_1^2, \dots, x_1^r; x_2^1, \dots, x_2^r; \dots; x_N^1, \dots, x_N^r,$$

т. е. векторы обучающей последовательности записываются подряд,

$$2) \ x_1^1, x_2^1, \dots, x_N^1; x_1^2, \dots, x_N^2; \dots; x_1^r, \dots, x_N^r,$$

т. е. сначала размещаются первые признаки всех N векторов, затем вторые и т. д.

Первый способ позволяет легко реализовать варианты с разным числом векторов во входной последовательности. Второй способ целесообразно использовать когда число векторов неизменно, а необходимо изменять число признаков, либо заменять одни признаки другими. Для большей универсальности программы следует накладывать ограничения не на N и r , а на их произведение Nr .

В целях экономии памяти можно размещать по несколько чисел в одной ячейке (если это позволяет допустимая точность представления чисел).

Признаки векторов могут быть представлены как в виде числовых, так и в виде логических переменных (присутствие или отсутствие некоторого признака).

На практике часто применяются признаки различной физической природы. Для их совместного использования иногда целесообразно преобразовать такие признаки в безразмерные нормированные величины. Существуют различные способы нормирования признаков. В качестве примера приведем два способа преобразования величин $x_j^i, i=1, \dots, r, j=1, \dots, N$:

$$1. \quad x_{j\text{норм}}^i = (x_j^i - x^i) / \sigma_i, \quad (9)$$

где $\bar{x}^i = (1/N) \sum_{j=1}^N x_j^i$ ($i=1, \dots, r$) — среднее значение i -го параметра;

$$\sigma_i = \left[(1/N) \sum_{j=1}^N (x_j^i - \bar{x}^i)^2 \right]^{1/2}$$

— среднеквадратичное отклонение i -го параметра для всех векторов

$$2. \quad x_{j_{\text{норм}}}^i = (x_j^i - x_{\min}^i) / (x_{\max}^i - x_{\min}^i), \quad (10)$$

где $x_{\min}^i = \min_j x_j^i$ — минимальное значение i -го параметра;

$x_{\max}^i = \max_j x_j^i$ — максимальное значение i -го параметра, $i=1, \dots, r$.

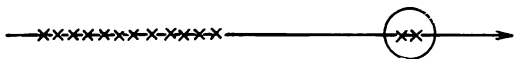


Рис. 3.

Недостатком второго способа является то, что при наличии «выбросов» значений признака происходит неоправданное сжатие диапазона значений этого признака. На рис. 3 изображена ось значений признака (крестиком отмечены конкретные значения, а кружком обведены выбросы).

В описанных способах нормирования влияние всех признаков на расстояние между векторами приблизительно одинаково; на практике часто оказывается, что признаки различаются по степени важности (имеют разные «веса»). Поэтому после нормирования следует учитывать весовые коэффициенты признаков, в общем случае не равные 1. Их значения определяются содержательной стороной задачи.

Настраиваемые параметры алгоритмов

Обычно с одной и той же входной последовательностью производят несколько просчетов, отличающихся набором настраиваемых параметров, предусмотренных

программой. Ниже перечисляются эти параметры: количество векторов N ; количество координат r (размерность пространства X); потенциальная функция $K(x, y)$; номер начального вектора в алгоритме «Спектр»; граничное значение $\Delta_{\text{гр}}$.

При практической реализации описанных алгоритмов хорошие результаты получены при использовании *потенциальной функции* вида

$$K(x, y) = [R(x, y)] = 1/[1 + \alpha R^p(x, y)], \quad (11)$$

где $R(x, y)$ — расстояние между векторами x и y .

Как указывалось выше, признаки векторов могут быть представлены в виде как числовых, так и логических переменных. В первом случае в качестве величины $R(x, y)$ удобно использовать евклидово расстояние

$$R(x, y) = R_e(x, y) = \left[\sum_{i=1}^r (x^i - y^i)^2 \right]^{1/2}.$$

Во втором случае удобно применять расстояние по Хеммингу: $R(x, y) = R_x(x, y)$, т. е. число несовпадающих разрядов в двоичных кодах векторов x, y .

Заметим, что для логических переменных x, y

$$R_x(x, y) = R_o^2(x, y).$$

Когда входные векторы характеризуются признаками, описываемыми как числовыми, так и логическими переменными, в качестве расстояния можно использовать величину

$$R(x, y) = R_x(\tilde{x}, \tilde{y}) + R_o^2(\hat{x}, \hat{y}),$$

где \tilde{x}, \tilde{y} — признаки, описываемые логическими переменными; \hat{x}, \hat{y} — признаки, описываемые числовыми переменными. Очевидно, что $x = (\tilde{x}, \hat{x})$, $y = (\tilde{y}, \hat{y})$.

Введем величину $R_{\text{ср}}$, определяемую равенством

$$\left. \frac{d^2 K [R(x, y)]}{dR^2(x, y)} \right|_{R_{\text{ср}}} = 0. \quad (12)$$

Отсюда видно, что $R_{\text{ср}}$ определяет точку перегиба потенциальной функции (11), изображенной на рис. 4, т. е. точку максимальной крутизны этой функции.

При описании алгоритма «Спектр» указывалось, что с его помощью строится последовательность \bar{K}_j , используемая для анализа структуры расположения входных векторов в пространстве X . Поэтому желательно, чтобы внутри группы значения \bar{K}_j , соответствующие одной и той же группы, были достаточно велики, а на границах между группами резко падали.

Для этого, например, достаточно выбрать $R_{\text{ср}}$ как среднее расстояние между векторами внутри группы.

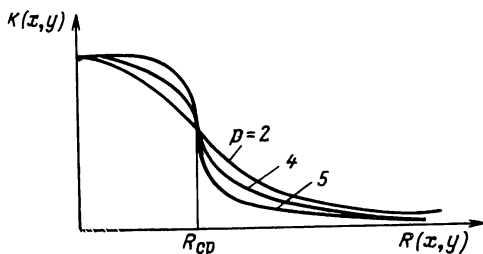


Рис. 4.

Параметр p в потенциальной функции (11) при фиксированном $R_{\text{ср}}$ характеризует крутизну этой функции в районе точки перегиба (рис. 4). Практически удобно брать $p=2t$, $t=1, 2, 3, \dots$

Параметр α в выражении (11) при известном $R_{\text{ср}}$ и фиксированном p определяется из функции (10):

$$\alpha = (p - 1) / (p + 1) R_{\text{ср}}^p.$$

Практически величину $R_{\text{ср}}$ оценить заранее обычно трудно. В этом случае можно воспользоваться следующим соображением: при $R < R_{\text{ср}}$, т. е. для векторов из одной группы (особенно для близких векторов из этой группы) значения $K(x, y)$ должны быть относительно велики. Поэтому значение α следует выбирать так, чтобы несколько первых членов последовательности \bar{K}_j имели значения порядка $0,7 \div 0,95$.

Практически нужное значение α подбирается за 2—4 пробы. Для этого в программе предусматривается просчет нескольких первых циклов с выводом этих значений \bar{K}_j на печать. После подбора значения α начинает работу программа «Спектр».

Номер вектора, с которого программа «Спектр» начинает строить последовательность $\{\tilde{x}_j\}$. Из описания алгоритма «Спектр» видно, что последовательность $\{\tilde{x}_j\}$, построенная в результате работы алгоритма, зависит от \tilde{x}_1 . Поэтому целесообразно делать несколько просчетов для обучающей последовательности с разными начальными векторами \tilde{x}_1 . Для удобства сравнения полученных результатов следует фиксировать нумерацию векторов. Вначале формируются номера векторов с помощью отдельной подпрограммы, затем, в процессе работы алгоритма номера переставляются одновременно с векторами.

Далее анализируются особенности работы программы «Объединение».

Выбор граничного значения $\Delta_{гр}$. При разбиении на ядра следует учитывать только $\Delta_j \geq 0$, поскольку члены $\Delta_j < 0$ соответствуют увеличению средней меры близости между векторами. Очевидно, что границы ядер должны приходиться на максимальные значения Δ_j . Отсюда непосредственно вытекает алгоритм разбиения последовательности $\{\tilde{x}_j\}$ на ядра: задаемся числом ядер a ; из последовательности $\{\Delta_j\}$ выбираем $(a-1)$ максимальных значений Δ_j . В качестве граничного значения $\Delta_{гр}$ принимается минимальное из них. Разбиение исходной последовательности на ядра A_1, \dots, A_a производится следующим образом:

- 1) вначале $x_1 \in A_1$;
- 2) пусть $\tilde{x}_{l-1} \in A$, тогда если $\Delta_l < \Delta_{гр}$, то $\tilde{x}_l \in A_1$; если $\Delta_l \geq \Delta_{гр}$, то $\tilde{x}_l \in A_2$ и, таким образом, построение ядра A_1 заканчивается (в него входят векторы $\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_{l-1}$);
- 3) остальные ядра строятся аналогично.

Описанный алгоритм разбиения на ядра обладает следующим недостатком. Во многих практических задачах между удаленными друг от друга группами точек существуют «мосты» (рис. 5). График изменения величины \tilde{K}_j в этом случае будет иметь вид, изображенный на рис. 6. Может оказаться, что в этом случае максимальное значение Δ_j не попадает на границу между группами. В таком случае этот алгоритм разбиения на ядра непригоден.

Далее описывается алгоритм разбиения на ядра, свободный от этого недостатка. Разбиение на ядра производится таким образом, чтобы в пределах каждого ядра разность между максимальным и минимальным значе-

нием K_j была бы меньше некоторой заданной величины, т. е.

1) $x_1 \in A_1$ $\delta_1 = 0$;

2) пусть $x_{l-1} \in A_1$, тогда:

а) если $\delta_{l-1} + \Delta_l \leq 0$, то $\tilde{x}_l \in A_1$ и $\delta_l = 0$;

б) если $0 < \delta_{l-1} + \Delta_l < \Delta_{\text{гр}}$, то $\tilde{x}_l \in A_1$ и $\delta_l = \delta_{l-1} + \Delta_l$;

в) если $\delta_{l-1} + \Delta_l \geq \Delta_{\text{гр}}$, то $\tilde{x}_l \in A_2$, $\delta_l = 0$ и таким образом построение ядра A_1 закончено;

3) аналогично строятся остальные ядра.

При использовании этого алгоритма разбиения последовательности $\{\tilde{x}_j\}$ число ядер заранее не определено.

Поэтому, если задано максимально возможное число ядер S_0 и получившееся число ядер $S > S_0$, то производится изменение параметра a , и разбиение на ядра производится снова.

Кроме того, второй способ

несколько сложнее первого при реализации его на ЦВМ.

Обработка особых ядер. Среди ядер, полученных в результате разбиения, могут оказаться ядра, состоящие из одного вектора. Назовем их *особыми*. Отличие этих ядер от остальных состоит в том, что величина $K(D, D)$ при $N_D = 1$ не определена [см. выражение (5)]. Поскольку величина $K(D, D)$ используется для получения классификации векторов, то необходимо, чтобы для всех ядер выполнялось неравенство $N_D > 1$. Для этого можно особые ядра вообще исключить из рассмотрения. При этом часть векторов не используется, что крайне нежелательно, особенно при малом числе векторов в обучающей последовательности. В этом случае можно присоединить каждое особое ядро A_i к такому ядру A_j , для которого

$$K(A_i, A_j) = \max_{p=1, \dots, S, p \neq i} K(A_i, A_p). \quad (13)$$

Формирование матрицы $\|K(A_i, A_j)\|$ и ее преобразование

Поскольку мера близости между двумя множествами точек A_i, A_j является симметричной функцией, т. е.

$$K(A_i, A_j) = K(A_j, A_i),$$

то достаточно вычислить элементы матрицы для $j \geq i$.

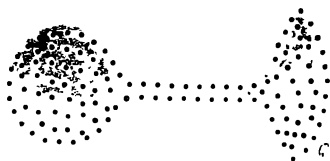


Рис. 5.

Согласно уравнениям (2) и (5) элементы матрицы вычисляются по формулам

$$K(A_i, A_j) = \frac{1}{n_{A_i} n_{A_j}} \sum_{x_p \in A_i} \sum_{x_t \in A_j} K(x_p, x_t), i \neq j, \quad (14)$$

$$K(A_i, A_i) = \frac{2}{n_{A_i}(n_{A_i} - 1)} \sum_{x_p \in A_i} \sum_{x_t \in A_i} K(x_p, x_t), t > p, \quad (15)$$

где n_{A_i} — число векторов в ядре A_i ; n_{A_j} — число векторов в ядре A_j .

Как видно, $K(A_i, A_j)$ есть мера близости между ядрами A_i и A_j , а $K(A_i, A_i)$ — средняя мера близости между векторами внутри ядра A_i .

Пусть на некотором шаге объединения ядра A_i и $A_j (i < j)$ объединяются в новое ядро \tilde{A}_i , которому присваивается номер, равный меньшему из чисел i, j ; число векторов в этом ядре:

$$n_{\tilde{A}_i} = n_{A_i} + n_{A_j}.$$

Очевидно, после объединения ядер A_i, A_j элементы матрицы должны быть преобразованы по формулам:

$$K(\tilde{A}_i, \tilde{A}_i) = [(n_{A_i} + n_{A_j})(n_{A_i} + n_{A_j} - 1)]^{-1} [n_{A_i}(n_{A_i} - 1) \times \\ \times K(A_i, A_i) + n_{A_j}(n_{A_j} - 1) K(A_j, A_j) + \\ + 2n_{A_i} n_{A_j} K(A_i, A_j)],$$

$$K(\tilde{A}_i, A_l) = \frac{1}{(n_{A_i} + n_{A_j})} [n_{A_i} K(A_i, A_l) + n_{A_j} K(A_j, A_l)], \\ l = 1, \dots, S; l \neq i.$$

Формулы (14) и (15) неудобны для программной реализации; целесообразнее было бы вычислять все элементы матрицы по общей формуле и делать необходимые поправки в формулах преобразования.

Ниже описываются два способа программной реализации алгоритма формирования и преобразования матрицы $\|K(A_i, A_j)\|$:

1. В первом способе формируется матрица $\|\hat{K}(A_i, A_j)\|$ ненормированных величин

$$\hat{K}(A_i, A_j) = \sum_{x_p \in A_i} \sum_{x_t \in A_j} K(x_p, x_t), i = 1, \dots, S; j \geq i. \quad (16)$$

При поиске ядер A_i, A_j , для которых величина $K(A_i, A_j)$, ($i \neq j$) максимальна, необходимо искать максимум среди величин

$$K(A_i, A_j) = \frac{1}{n_{A_i} n_{A_j}} \hat{K}(A_i, A_j); j > i. \quad (17)$$

Формулы преобразования в этом случае примут вид:

$$\hat{K}(\tilde{A}_i, \tilde{A}_i) = \hat{K}(A_i, A_i) + \hat{K}(A_j, A_j) + 2\hat{K}(A_i, A_j); \quad (18)$$

$$\hat{K}(\tilde{A}_i, A_l) = \hat{K}(A_i, A_l) + \hat{K}(A_j, A_l). \quad (19)$$

Такой способ формирования и преобразования матрицы $\|K(A_i, A_j)\|$ достаточно просто реализуется на ЦВМ. Недостатком этого способа является необходимость вычисления величин (17) в процессе поиска максимума $K(A_i, A_j)$ на каждом шаге объединения.

Кроме того, некоторые из величин $K(A_i, A_j) > 1$. Для ряда вычислительных машин (например, типа «Урал») в целях экономии объема оперативной памяти необходимо, чтобы все числа были меньше единицы.

2. Во втором способе формируется $\|\bar{K}(A_i, A_j)\|$ — матрица, элементы которой вычисляются по общей формуле

$$\bar{K}(A_i, A_j) = \frac{1}{n_{A_i} n_{A_j}} \sum_{x_p \in A_i} \sum_{x_t \in A_j} K(x_p, x_t), j \geq i. \quad (20)$$

Заметим, что для $i \neq j$ $\bar{K}(A_i, A_j) = K(A_i, A_j)$.

Формулы преобразования элементов матрицы в этом случае имеют вид

$$\bar{K}(\tilde{A}_i, \tilde{A}_i) = \frac{1}{(n_{A_i} + n_{A_j})^2} [n_{A_i}^2 \bar{K}(A_i, A_i) + 2n_{A_i} n_{A_j} \bar{K}(A_i, A_j) + n_{A_j}^2 K(A_j, A_j)];$$

$$\bar{K}(\tilde{A}_i, A_l) = \frac{1}{n_{A_i} + n_{A_j}} [n_{A_i} \bar{K}(A_i, A_l) + n_{A_j} \bar{K}(A_j, A_l)], l \neq i, j.$$

Недостатком этого способа является то, что в процессе формирования и преобразования матрицы необходимо использовать величины n_{A_i}, n_{A_j} .

При преобразовании матрицы удобно, чтобы на каждом шаге для не объединенных ядер сохранялись старые номера и место размещения в МОЗУ соответствующих

элементов. Для того чтобы ядро A_j после его объединения с ядром A_i не участвовало в процедуре подсчета максимума $K(A_l, A_p)$, т. е. нахождения следующей пары ближайших ядер, необходимо в ячейки, в которых хранятся величины $K(A_j, A_l)$, $l=1, \dots, S$, заслать нули. Опыт показывает, что после объединения ядер A_i, A_j не следует «сжимать» матрицу $\|K(A_l, A_p)\|$, ибо это существенно усложняет программу.

4. ОЦЕНКА ПОЛУЧЕННЫХ КЛАССИФИКАЦИЙ. ВЫБОР КРИТЕРИЯ КАЧЕСТВА КЛАССИФИКАЦИИ

При использовании алгоритмов «Спектр» и «Объединение» в практических задачах хорошо себя зарекомендовали два критерия качества классификации (8):

$$J = J_1 - J_2, \quad (21)$$

$$J = (J_1 - J_2) / (J_1 + J_2). \quad (22)$$

Поскольку величины J_1 и J_2 характеризуют структуру входной последовательности, рекомендуется на каждом шаге объединения выводить на печать как значения J , так и значения J_1 и J_2 .

При обработке результатов работы алгоритма «Объединение» удобно строить графики $J(k)$, где k — число классов на $(S-k)$ -м шаге объединения.

Следует иметь в виду, что окончательный анализ полученных классификаций должен проводиться с учетом конкретной задачи. При этом может оказаться, что либо выбранной классификации соответствует локальный (а не абсолютный) максимум величины J (8), либо вообще значение J , соответствующее этой классификации, не является экстремальным. Однако обычно хорошим классификациям отвечают достаточно высокие значения J . Именно такие классификации следует отбирать для детального анализа.

При подсчете величин J_1 и J_2 , входящих в (21), (22), следует учитывать способ формирования матрицы.

1. Если элементы матрицы вычисляются по формулам (14), (15), то J_1, J_2 подсчитываются непосредственно по (6), (7).

2. Если элементы матрицы вычисляются по формуле (16), т. е. формируется матрица $\|\hat{K}(A_i, A_j)\|$, то выраже-

ния (6), (7), принимают вид

$$J_1 = \frac{1}{N} \sum_{i=1}^S \frac{1}{n_{A_i} - 1} [\hat{K}(A_i, A_i) - n_{A_i}],$$

$$J_2 = \frac{1}{n_1} \sum_{j>i}^S \hat{K}(A_i, A_j),$$

где N — общее число объектов, а $n_1 = \sum_{i=1}^S \sum_{j>1}^S n_{A_i} n_{A_j}$.

3. Если элементы матрицы вычисляются по формуле (20), т. е. формируется матрица $\|\hat{K}(A_i, A_j)\|$, то выражение (7) не меняется, а (6) принимают вид

$$J_1 = \frac{1}{N} \sum_{i=1}^S \frac{n_{A_i}}{n_{A_i} - 1} [n_{A_i} \hat{K}(A_i, A_i) - 1].$$

Печать и обработка результатов

При работе описанных алгоритмов рекомендуется предусмотреть возможность вывода на печать следующих результатов:

1. В программе «Спектр»: последовательность $\{\hat{K}_j\}$, номера векторов.

2. В программе «Объединение»:

а) номера ядер и номера векторов каждого ядра после разбиения входной последовательности по $\Delta_{\text{гр}}$ и после присоединения особых ядер (если такое присоединение проводится);

б) номера объединяемых ядер и порядок матрицы на каждом шаге объединения;

в) величины J_1, J_2, J ;

г) элементы матрицы $\|K(A_i, A_j)\|$.

Поскольку для больших S матрица $\|K(A_i, A_j)\|$ содержит значительное число элементов, удобно печатать элементы этой матрицы не на каждом шаге объединения, а только на тех, которые представляют интерес для анализа.

При анализе полученной классификации величины \hat{K}_j, J_1, J_2, J целесообразно представить в виде графиков. При построении графика \hat{K}_j удобно пользоваться неравномерной логарифмической шкалой по оси ординат. На-

пример, можно выбрать следующую шкалу по оси \tilde{K}_j : пусть порядок минимального числа в последовательности $\{\tilde{K}_j\}$ равен $(-v)$; тогда ось ординат разбивается на v одинаковых отрезков, внутри которых шкала равномерна. На рис. 6 дана такая шкала для $v=3$.

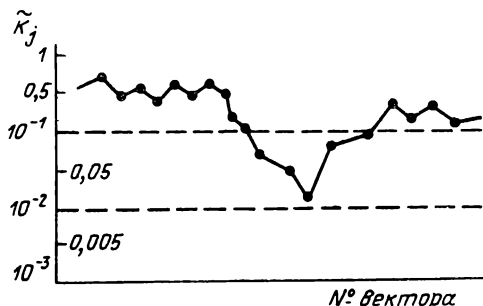


Рис. 6.

6. ТЕСТЫ ДЛЯ ОТЛАДКИ ПРОГРАММ

Отлаживать программы удобно на тестовой последовательности, для которой заранее известны (либо легко просчитываются вручную) результаты на каждом этапе вычислений. Тестовая последовательность может вводиться извне, как любая входная последовательность, либо формироваться специальной программой. В эту программу, кроме блока формирования тестовой последовательности, можно ввести указания о выводе на печать промежуточных и конечных результатов, о проверке программ в различных режимах и т. д. Тест не может быть слишком простым: например, нельзя, чтобы в результате обработки его программой «Спектр» порядок векторов остался неизменным. Кроме того, длина тестовой последовательности не может быть слишком большой, ибо время ее обработки существенно зависит от произведения Nr , где N — число векторов в тестовой последовательности, r — размерность вектора.

СПИСОК ЛИТЕРАТУРЫ

1. Дорофеев А. А. Алгоритмы обучения машин распознаванию образов без учителя, основанные на методе потенциальных функций. «Автоматика и телемеханика», 1966, № 10.
2. Айзерман М. А., Браверман Э. М., Розоноэр Л. И. Теоретические основы метода потенциальных функций в задаче обучения автоматов разделению входных ситуаций на классы. «Автоматика и телемеханика», 1964, т. XXV, № 6.

УДК 62—5:007:621.391:519.2

Машины, обучающиеся распознаванию образов. Вапник В. Н. Сб. «Алгоритмы обучения распознаванию образов» под ред. В. Н. Вапника. Изд-во «Советское радио», 1972, стр. 5.

Рассматриваются общие теоретические вопросы анализа и синтеза алгоритмов обучения распознаванию образов. Исследуется сходимость каждого алгоритма, приведенного в сборнике.

Рис. 3, библи. 3 назв

УДК 62—5:007:621.391:519.2

Обучающиеся опознающие системы и рекуррентные конечно-сходящиеся алгоритмы. Фомин В. Н., Якубович В. А. Сб. «Алгоритмы обучения распознаванию образов» под ред. В. Н. Вапника. Изд-во «Советское радио», 1972, стр. 29.

Приводятся три рекуррентных конечно-сходящихся алгоритма построения гиперплоскости, разделяющей два множества векторов.

Библи. 1 назв.

УДК 62—5:007:621.391:519.2

Рекуррентный алгоритм разделения двух множеств. Козинец Б. Н. Сб. «Алгоритмы обучения распознаванию образов» под ред. В. Н. Вапника. Изд-во «Советское радио», 1972, стр. 43.

Приводится алгоритм построения разделяющей гиперплоскости, основанный на поиске направления, определяющего расстояние между выпуклыми оболочками множеств.

Рис. 1, библи. 2 назв.

УДК 62—5:007:621.391:519.2

Программы для решения задач распознавания образов методом потенциальных функций. Мучник И. Б., Петренко Е. С. Сб. «Алгоритмы обучения распознаванию образов» под ред. В. Н. Вапника. Изд-во «Советское радио», 1972, стр. 50.

Описывается система программ, реализующая метод потенциальных функций в задачах обучения распознаванию образов и восстановления функций.

Библи. 6 назв.

УДК 62—5:007:621.391:519.2

Об одном рекуррентном алгоритме нахождения плоскости, разделяющей выпуклые множества на единичном кубе. Фомин В. Н. Сб. «Алгоритмы обучения распознаванию образов» под ред. В. Н. Вапника. Изд-во «Советское радио», 1972, стр. 85.

Приводится рекуррентный алгоритм построения разделяющей гиперплоскости, направляющие коэффициенты которой могут принимать только два значения — нуль и единица.

Библи. 5 назв.

УДК 62—5:007:621.391:519.2

Алгоритмы обучения распознаванию образов, использующие метод «обобщенного портрета». Алгоритмы ОП-1, ОП-2, ОП-3. Вапник В. Н., Журавель А. А., Червоненкис А. Я., Сб. «Алгоритмы обучения распознаванию образов» под ред. В. Н. Вапника. Изд-во «Советское радио», 1972, стр. 89.

Описывается система алгоритмов, реализующих метод «обобщенного портрета». С помощью алгоритма ОП-1 может быть построена разделяющая гиперплоскость для множеств, выпуклые оболочки которых пересекаются. Алгоритм ОП-3 предназначен для построения кусочно-линейных разделяющих поверхностей.

Рис. 2, библи. 2 назв.

УДК 62—5:007:621.391:519.2

Алгоритм обучения распознаванию образов «Кора». Вайнцвайг М. Н. Сб. «Алгоритмы обучения распознаванию образов» под ред. В. Н. Вапника. Изд-во «Советское радио», 1972, стр. 110.

Описывается алгоритм обучения распознаванию образов, основанный на отборе для каждого класса признаков, достаточных на множестве примеров. Такие признаки отбираются среди всех возможных конъюнкций длины не больше трех среди исходных параметров (или их отрицаний).

Библ. 1 назв.

УДК 62—5:007:621.391:519.2

Программа классификации многопараметрических объектов, основанная на отборе статистически полезных признаков. Браиловский В. Л., Лунц А. Л. Сб. «Алгоритмы обучения распознаванию образов» под ред. В. П. Вапника. Изд-во «Советское радио», 1972, стр. 116.

Описывается алгоритм обучения распознаванию образов, основанный на отборе информационных подпространств исходного пространства параметров, в каждом из которых восстанавливается разделяющая функция. Из множества таких разделяющих функций конструируется решающее правило.

Рис. 6, библиография 4 назв.

УДК 62—5:007:621.391:519.2

Алгоритмы обучения распознаванию образов, использующие метод «обобщенного портрета». Алгоритм ОП-4, ОП-5, ОП-6, ОП-7. Вапник В. Н., Глазкова Т. Г., Червоненкис А. Я. Сб. «Алгоритмы обучения распознаванию образов» под ред. В. Н. Вапника. Изд-во «Советское радио», 1972, стр. 136.

Описывается система алгоритмов, реализующих метод «обобщенного портрета». С помощью описанных алгоритмов может быть построена оптимальная разделяющая гиперплоскость, найдено подпространство минимальной размерности, в котором заданные два множества векторов могут быть разделены гиперплоскостью, построена кусочно-линейная разделяющая поверхность.

УДК 62—5:007:621.391:519.2

Некоторые алгоритмы медицинской диагностики и прогнозирования. Баталова З. С., Куренков Е. Ф., Неймарк Ю. И., Образцова Н. Д. Сб. «Алгоритмы обучения распознаванию образов» под ред. В. Н. Вапника. Изд-во «Советское радио», 1972, стр. 150.

Описывается система алгоритмов обучения распознаванию образов, основанная на преобразовании исходного пространства и построения «простых» решающих правил в преобразованном пространстве.

Рис. 6, библиография 3 назв.

УДК 62—5:007:621.391:519.2

Реализация алгоритмов обучения распознаванию образов без учителя на ЦВМ. Дорофеюк А. А., Лумельский В. Я. Сб. «Алгоритмы обучения распознаванию образов» под ред. В. Н. Вапника. Изд-во «Советское радио», 1972, стр. 181.

Приводятся три алгоритма автоматической классификации объектов (обучение распознаванию образов без учителя). Описываются программы, реализующие эти алгоритмы.

Рис. 6, библиография 2 назв.

СОДЕРЖАНИЕ

| | |
|--|-----|
| Предисловие | 3 |
| Вапник В. Н. Машины, обучающиеся распознаванию образов | 5 |
| Фомин В. Н., Якубович В. А. Обучаемые опознающие системы и рекуррентные конечно-сходящиеся алгоритмы | 29 |
| Козинец Б. Н. Рекуррентный алгоритм разделения выпуклых оболочек двух множеств | 43 |
| Мучник И. Б., Петренко Е. С. Программы для решения задач распознавания образов методом потенциальных функций | 50 |
| Фомин В. Н. Об одном рекуррентном алгоритме нахождения гиперплоскости, разделяющей выпуклые множества на единичном кубе | 85 |
| Вапник В. Н., Журавель А. А., Червоненкис А. Я. Алгоритмы обучения машин распознаванию образов ОП-1, ОП-2, ОП-3 | 89 |
| Вайнцвайг М. Н. Алгоритм обучения распознаванию образов «Кора» | 110 |
| Браиловский В. Л., Луц А. Л. Программа обучения распознаванию, основанная на отборе статистически полезных признаков | 116 |
| Вапник В. Н., Глазкова Т. Г., Червоненкис А. Я. Алгоритмы обучения машин распознаванию образов ОП-4, ОП-5, ОП-6, ОП-7 | 136 |
| Баталова З. С., Куренков Е. Ф., Неймарк Ю. И., Образцова Н. Д. Некоторые алгоритмы медицинской диагностики и прогнозирования | 150 |
| Дорофеюк А. А., Лумельский В. Я. Реализация алгоритмов обучения распознаванию образов «без учителя» на ЭВМ | 181 |

55 коп.