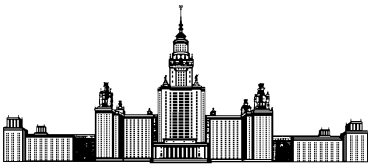
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА СТУДЕНТА 417 ГРУППЫ

«Обработка множеств логических закономерностей с помощью дисперсионного критерия»

Выполнил:
студент 4 курса 417 группы
Лисяной Александр Евгеньевич

Научный руководитель:
д.ф-м.н., профессор
Рязанов Владимир Васильевич

Заведующий кафедрой
Математических Методов
Прогнозирования, академик РАН _____ Ю. И. Журавлёв

К защите допускаю
«_____» _____ 2015 г.

К защите рекомендую
«_____» _____ 2015 г.

Москва, 2015

Содержание

Аннотация

Одной из основных задач машинного обучения является задача классификации. Существует множество разнообразных алгоритмов классификации, в том числе алгоритмы, основанные на поиске логических закономерностей в данных. Этот поиск — довольно трудоемкая задача. Приходится прибегать к различного рода эвристикам, то есть как-то ограничивать время работы полного перебора. Полученные логические закономерности могут оказаться слишком сложными (переобученными) или очень похожими (не *диверсифицированными*). Поэтому, как правило, применяются различные методы обработки множества логических закономерностей. Целью данной работы является реализовать и применить метод обработки множества логических закономерностей с помощью дисперсионного критерия к результату работы алгоритма «логические закономерности», реализованного в системе «Распознавание».

1 Введение

Одной из основных задач машинного обучения является задача классификации. Существует множество разнообразных алгоритмов классификации, в том числе алгоритмы, основанные на поиске логических закономерностей в данных. Отличительной чертой таких алгоритмов является то, что полученные в результате работы логические закономерности можно, как правило, *проинтерпретировать*, *оценить* и *обработать*.

Поиск логических закономерностей в данных — довольно трудоемкая задача. Приходится прибегать к различного рода эвристикам, то есть как-то ограничивать время работы полного перебора. Это ведет к тому, что получаемые множества логических закономерностей не являются гарантированно наилучшими. Более того, полученные логические закономерности могут оказаться слишком сложными (переобученными) или очень похожими (не *диверсифицированными*). Поэтому, как правило, применяются различные методы обработки множества логических закономерностей.

Целью данной работы является реализовать и применить метод обработки множества логических закономерностей с помощью дисперсионного критерия к результату работы алгоритма «логические закономерности», реализованного в системе «Распознавание». Предлагается использовать описание логических закономерностей с помощью вектора левых и правых границ и сравнить его с бинаризованным описанием логических закономерностей, предложенном в [11]. На шаге кластеризации в дисперсионном критерии предлагается использовать «вес» для отделения более информативных правил.

Обработанные множества логических закономерностей сравниваются как между собой (в ходе обработки используются различные способы описания правил и критерии информативности), так и с исходным множеством логических закономерностей, полученным в результате настройки алгоритма системы «Распознавания». В результате предложенный в данной работе подход с использованием вектора левых и правых границ и взвешенной кластеризации в целом работает лучше, чем подход с бинаризованным описанием, изложенный в [11]. Качество классификации сравнимо с исходным алгоритмом системы «Распознавание», но при этом размер множества логических закономерностей меньше.

1.1 Определения и обозначения

Основные определения и обозначения будем вводить согласно [12]. Пусть имеется пространство объектов X и конечное множество имен классов $Y = \{1, \ldots, M\}$. Пусть также имеется *обучающая выборка* $X^l = (x_i, y_i)_{i=1}^l$, в которой для каждого объекта x_i известен его класс $y_i \in Y$. Для восстановления целевой зависимости $y^*(x_i) = y_i$ построим алгоритм классификации $a\colon X \to Y$, аппроксимирующий y^* на всём пространстве объектов X.

Для решения такой задачи классификации (восстановления зависимости y^* по обучающей выборке X^l) существует множество разнообразных алгоритмов. Некоторые из них основаны на поиске логических закономерностей.

Определение 1 *Функция* $\varphi(x)\colon X \to \{0, 1\}$ *называется* предикатом. *Говорят, что предикат* φ выделяет *или* покрывает *объект* $x \in X$, *если* $\varphi(x) = 1$.

Определение 2 *Предикат называется* закономерностью, *если он выделяет достаточно много объектов своего класса и практически не выделяет объекты чужого класса.*

Понятие *закономерности* с одной стороны интуитивно, с другой стороны слишком неточно с математической точки зрения. Формализуем его как в [12].

- Пусть P_c — число объектов класса c в выборке X^l.
- Пусть $p_c(\varphi)$ — число объектов x класса c, на которых $\varphi(x) = 1$
- Пусть N_c — число объектов всех остальных классов $Y \setminus \{c\}$
- Пусть $n_c(\varphi)$ — число объектов x не из объектов класса c, на которых $\varphi(x) = 1$
- Обозначим долю негативных объектов среди всех объектов, выделяемых *закономерностью*, через $E_c(\varphi, X^l) = \frac{n_c(\varphi)}{p_c(\varphi)+n_c(\varphi)}$
- Обозначим долю позитивных объектов, выделяемых *закономерностью*, через $D_c(\varphi, X^l) = \frac{p_c(\varphi)}{l}$

Определение 3 *Предикат* $\varphi(x)$ *называется* логической ε,δ-закономерностью *для класса* $c \in Y$, *если* $E_c(\varphi, X^l) \leq \varepsilon$ *и* $D_c(\varphi, X^l) \geq \delta$ *при заданных достаточно малом* ε *и достаточно большом* δ *из отрезка* $[0, 1]$

Логические ε,δ-закономерности также называют *логическими закономерностями*, *закономерностями* или *правилами*.

Вообще говоря, предикаты $\varphi(x)\colon X \to Y$ могут иметь довольно сложный вид. Например, если в качестве объектов из X выступают точки на числовой прямой, то для классификации на рациональные и иррациональные числа можно взять аналог функции Дирихле в качестве предиката:

$$\varphi(x) = \begin{cases} 1 & x \in \mathbf{Q} \\ 0 & x \in \mathbf{R} \setminus \mathbf{Q} \end{cases}$$

Для того, чтобы предикаты были более простыми, введем ограничение на их вид так, как было сделано в [13].

Определение 4 *Пусть $c_1, c_2 \in R$, $f_j(x)$ — j-ый признак объекта x, тогда* параметрическим элементарным предикатом *будем называть*

$$P^{c_1,c_2}(f_j(x)) = \begin{cases} 1 & \textit{при } c_1 \le f_j(x) \le c_2 \\ 0 & \textit{иначе} \end{cases}$$

Такой предикат называется элементарным, потому что в нем используется лишь один признак из всего признакового описания объекта. Параметрическим же он называется потому, что параметры $c_1, c_2 \in R$ для каждого отдельного признака $f_j(x)$ выбираются, вообще говоря, свои.

Будем считать, что *логическая ε,δ-закономерность* $\varphi(x)$ имеет вид конъюнкции параметрических элементарных предикатов:

$$\varphi(x) = \wedge_{f_j(x)} P^{c_1^j,c_2^j}(f_j(x)) \tag{1}$$

Здесь индексы j над параметрами c_1^j, c_2^j означают, что для каждого признака могут быть выбраны свои границы. Для логических закономерностей такого вида есть простая геометрическая интерпретация — это построенные по объектам обучающей выборки X прямоугольные гиперпараллелепипеды.

2 Проблема поиска логических закономерностей

Поиск наиболее информативных логических закономерностей требует полного перебора. Пусть множество $\mathcal{P}$ параметрических элементарных предикатов конечно. Тогда множество $\mathcal{K}_K$ логических закономерностей, состоящих не более, чем из K термов, имеет вид:

$$\mathcal{K}_K = \{\varphi(x) = P_1(x)\&P_2(x)\&\ldots\&P_k(x) \mid P_1, \ldots, P_k \in \mathcal{P}, k \leq K\}$$

Количество логических закономерностей: $|\mathcal{K}| = |\mathcal{P}|^K$. Это слишком много для перебора, поэтому используют различные эвристики для его сокращения.

2.1 Алгоритмы по созданию логических закономерностей

К таким алгоритмам относятся «KOPA» [10], «IREP» и «RIPPER» [4], «SLIPPER» [1], «ID3» [5], «C4.5rules» [3], [2]. В качестве классического примера такого алгоритма можно привести двухклассовую версию «IREP» [1].

Algorithm 1: Incremental Reduced Error Pruning (IREP)

Вход : Pos — множество элементов класса $+1$
Neg — множество элементов класса -1
Выход: Rules — множество правил
Функция `IREP(`*Pos, Neg*`)`
  Завести пустой набор правил Ruleset;
  Пока *Pos не опустеет*
    `/* Нужно обучить и упростить очередное правило Rule */`
    Разделить (Pos, Neg) на (GrowPos, GrowNeg) и (PrunePos, PruneNeg);
    Обучить правило Rule по паре множеств (GrowPos, GrowNeg);
    Упростить правило Rule по паре множеств (PrunePos, PruneNeg);
    Если *доля ошибок Rule на (PrunePos, PruneNeg) больше 50%* **то**
      Вернуть *Ruleset*;
    Иначе
      Добавить правило Rule в Ruleset;
      Убрать из (Pos, Neg) объекты, которые покрывает правило Rule;
  Вернуть *Ruleset*;

Алгоритм IREP [1] является одним из классических примеров жадных алгоритмов построения правил, работающий по принципу «разделяй и властвуй» (divide and conquer). Этот принцип проявляется в том, что правила создаются по одному, а объекты, которые покрываются созданным правилом,

исключаются из рассмотрения. Многие алгоритмы («КОРА», «ТЭМП» [12]) по смыслу очень похожи и отличаются только критериями останова и способами упрощения полученных правил.

2.2 Проблемы алгоритмов поиска логических закономерностей

Необходимость полного перебора и эвристическая природа алгоритмов поиска логических закономерностей ставит несколько проблем:

1. Невозможно гарантировать получение наилучшего набора правил.
2. Получаемые правила зачастую нуждаются в упрощении.
3. Получаемые правила зачастую нуждаются в диверсификации [12], [14].

Избавиться от первой проблемы невозможно, с ней можно только бороться с помощью различных эвристических алгоритмов. Для решения второй проблемы существует большое число подходов, обзор можно посмотреть в [7]. Проблема диверсификации, наконец, заключается в том, что довольно часто находятся очень похожие закономерности. На примере из [12] можно понять, почему это действительно плохо:

Пример 1 *Предположим, что в некоторой экспертной комиссии есть два эксперта, мнения которых всегда (или почти всегда) совпадают по всем вопросам. Интуитивно понятно, что если убрать одного из экспертов, то качество работы новой комиссии будет таким же (или почти таким же).*

3 Обработка логических закономерностей

Для того, чтобы упростить и диверсифицировать полученное в результате работы некоторого логического алгоритма классификации множества логических закономерностей $\varphi_1(x), \ldots, \varphi_n(x)$, предлагается провести его обработку с помощью дисперсионного критерия. Основную схему этого подхода, предложенного В.В. Рязановым, можно описать следующим образом:

1. Каждой логической закономерности φ_j^i из множества $\mathcal{K}_i = \{\varphi_1^i(x), \ldots, \varphi_t^i(x)\}$ логических закономерностей, покрывающих класс y_i, поставить в соответствие некоторый вектор $\boldsymbol{z}_j$, описывающий эту закономерность.

2. Провести кластеризацию векторов $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_t$ на $k \leq t$ кластеров и найти центры этих кластеров $\boldsymbol{z}_1^*, \ldots, \boldsymbol{z}_k^*$.

3. По центрам кластеров восстановить $\varphi_1^*(x), \ldots, \varphi_k^*(x)$ — некоторые, вообще говоря новые, логические закономерности для класса y_i.

3.1 Способы представления логических закономерностей

В этой работе рассмотрено два способа представить логические закономерности: с помощью бинарного вектора и с помощью вектора левых и правых границ.

Представление с помощью бинарного вектора было исследовано в [11]. Идея заключается в том, чтобы каждой логической закономерности $\varphi(x)$ поставить в соответствие вектор $\boldsymbol{z} \in \{0,1\}^l$ так, что

$$\boldsymbol{z}_i = \begin{cases} 1 & \text{если } \varphi(x_i) = 1 \\ 0 & \text{иначе} \end{cases}$$

В данной работе предлагается использовать представление с помощью вектора левых и правых границ, которое использует введенное в 1.1 представление логической закономерности $\varphi(x)$ в виде параметрических элементарных предикатов (1). При таком подходе вектор $\boldsymbol{z}$ принимает вид:

$$\boldsymbol{z} = (c_1^1, c_2^1, c_1^2, c_2^2, \ldots, c_1^d, c_2^d)$$

3.2 Алгоритм кластеризации логических закономерностей

На этом шаге нужно разбить полученное описание множества логических закономерностей $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_t$ на $k \leq t$ непересекающихся классов $\boldsymbol{S} = \{S_1, S_2, \ldots, S_k\}$ таким образом, чтобы минимизировать функционал 2. При этом для каждой закономерности предлагается учитывать «вес», то есть добавить константные коэффициенты β_j:

$$\boldsymbol{S}^* = \arg\min_{\boldsymbol{S}} \sum_{i=1}^{k} \sum_{\boldsymbol{z}_j \in S_i} \beta_j \|\boldsymbol{z}_j - \boldsymbol{\mu}_i\|^2 \tag{2}$$

Тогда алгоритм кластеризации (алгоритм Ллойда [6], алгоритм K-средних [8]) примет вид [2]

Algorithm 2: Алгоритм кластеризации (Ллойда, K-средних)

Вход : Описание закономерностей $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_t$

Выход: $\boldsymbol{S}^* = S_1, \ldots, S_k$ — разбиение на непересекающиеся кластеры

Функция *Кластеризовать*$(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_t)$

Инициализировать центры кластеров $\boldsymbol{\mu}_1^1, \ldots, \boldsymbol{\mu}_k^1$;

Пока *кластеризация не стабилизируется*

`/* распределить объекты по кластерам, при этом */`

`/* минимизировать функционал качества(2) */`

$S_i^t = \left\{\boldsymbol{z}_p : \beta_p \|\boldsymbol{z}_p - \boldsymbol{\mu}_i^t\|^2 \leq \beta_p \|\boldsymbol{z}_p - \boldsymbol{\mu}_j^t\|^2, \forall j : 1 \leq j \leq k\right\}$;

`/* пересчитать центры кластеров с учетом веса */`

$\boldsymbol{\mu}_i^{t+1} = \frac{1}{|S_i^t|} \frac{\sum_{\boldsymbol{z}_j \in S_i^t} \beta_j \boldsymbol{z}_j}{\sum_{\boldsymbol{z}_j \in S_i^t} \beta_j}$;

Вернуть $S_1^t, \ldots, S_k^t$;

3.3 Восстановление логических закономерностей

После того, как алгоритм кластеризации находит разбиение $\boldsymbol{S}^*$, необходимо по центрам кластеров $\hat{\boldsymbol{z}}_1^*, \ldots, \hat{\boldsymbol{z}}_k^*$ восстановить логические закономерности $\varphi_1^*(x), \ldots, \varphi_k^*(x)$.

В случае, когда логические закономерности исходно были представлены в виде бинарных векторов, полученный центр кластера $\hat{\boldsymbol{z}}_i^*$ уже, вообще говоря, не является бинарным вектором. Поэтому для каждого вектора $\hat{\boldsymbol{z}}_i^*, i = 1, \ldots, k$ выбирается некоторый порог бинаризации θ_i таким образом, чтобы логическая закономерность $\varphi_i^*(x)$, отвечающая представлению:

$$\boldsymbol{z}_i^* = \begin{cases} 1 & \text{если } \hat{\boldsymbol{z}}_i^* \geq \theta_i \\ 0 & \text{иначе} \end{cases}$$

была наиболее информативной среди всех рассмотренных значений порога θ_i. Информативность логической закономерности рассчитывается с помощью известных критериев информативности, таких как энтропийный критерий IGain. Подробный обзор использования разных критериев для этой задачи представлен в [11].

В случае, когда логические закономерности исходно описывались с помощью вектора левых и правых границ, полученный центр кластера $\hat{\boldsymbol{z}}_i^*$ можно использовать в качестве описания $\boldsymbol{z}_i^*$ некоторой новой логической закономерности. То есть нет необходимости проводить отдельную процедуру по восста-

новлению описания.

4 Вычислительные эксперименты

Основной целью вычислительных экспериментов было реализовать подход обработки множества логических закономерностей с помощью дисперсионного критерия, описанного в разделе 3. Также целью вычислительного эксперимента было сравнить способы представления логических закономерностей, описанные в разделе 3.1. Для этого была принята следующая схема проведения эксперимента:

1. Исходная выборка делилась пополам на подвыборки обучения и контроля. При этом для подвыборок сохранялись одинаковые пропорции классов.

2. На подвыборке обучения в системе «Распознавание» [15] настраивался классификатор «логические закономерности».

3. Логические закономерности настроенного классификатора импортировались в программу, реализующую дисперсионный подход в обработке логических закономерностей.

4. По этому набору логических закономерностей с помощью алгоритма «простого голосования» [12] проводилась классификация контрольной подвыборки.

5. Далее проводилось описание логических закономерностей одним из двух способов из раздела 3.1. Полученные описания проходили кластеризацию на $t = 2, 3, \ldots, n, \ldots$ кластеров.

6. По полученным на очередном шаге t центрам кластеров восстанавливались новые логические закономерности, которые затем использовались в алгоритме «простого голосования» для классификации контрольной подвыборки.

7. Полученные значения доли правильно классифицированных объектов для разных способов представления и для разного числа кластеров t выносилось на график.

4.1 Исходные данные и условия эксперимента

Исходные данные были взяты из репозитория UCI [9]. Ниже представлена сводная таблица по использованным данным [1]:

Выборка	Объекты	Классы	Признаки
Iris[1]	150	50/50/50	4
Wine[2]	178	59/71/48	13
Climate[3]	540	46/494	11
Ionosphere[4]	351	126/255	34

Таблица 1: Сводная таблица по использованным данным

Выборка Iris[1] ставит задачу классификации растений семейства ирисов. Выборка Wine[2] содержит данные химического анализа вина, полученного от трех разных итальянских виноделов. Выборка Climate[3] содержит информацию об удачных и неудачных запусках симулирования погоды. Наконец, выборка Ionosphere[4] исследует свободные электроны в ионосфере.

4.2 Результаты эксперимента

Результаты запусков различных конфигураций метода обработки множества логических закономерностей с помощью дисперсионного критерия представлены на графиках 1a, 1b, 2a, 2b, 3a и 3b. Сплошная линия красного цвета обозначает качество классификации исходного множества логических закономерностей. Синяя линия с кружками обозначает предложенный в данной работе подход, зеленая линия с треугольниками и бирюзовая с квадратами — подходы с бинаризованным описанием и критериями информативности IGain и Stat [12].

Критерий информативности IGain — это энтропийный критерий информативности правила $\varphi(x)$, отвечающего классу y_i, который имеет вид (3)

[1] `http://archive.ics.uci.edu/ml/datasets/Iris`
[2] `http://archive.ics.uci.edu/ml/datasets/Wine`
[3] `https://archive.ics.uci.edu/ml/datasets/Climate+Model+Simulation+Crashes`
[4] `https://archive.ics.uci.edu/ml/datasets/Ionosphere`

$$\text{IGain} = H\left(\frac{P}{P+N}, \frac{N}{P+N}\right) - \frac{p+n}{P+N} H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \frac{P+N-p-n}{P+N} H\left(\frac{P-p}{P+N-p-n}, \frac{N-n}{P+N-p-n}\right) \quad (3)$$

В формуле (3) введено обозначение энтропии $H(p,q) = -p\log_2(p) - q\log_2(q)$, P, N — число объектов класса y_i и не y_i соответственно, а p, n — число объектов из класса y_i и не из класса y_i, которые покрывает логическая закономерность $\varphi(x)$.

Критерий Stat имеет вид (4)

$$\text{Stat} = -\ln \frac{C_{P_1}^{p_1} \dots C_{P_K}^{p_K}}{C_l^{p_1+\dots+p_K}} \quad (4)$$

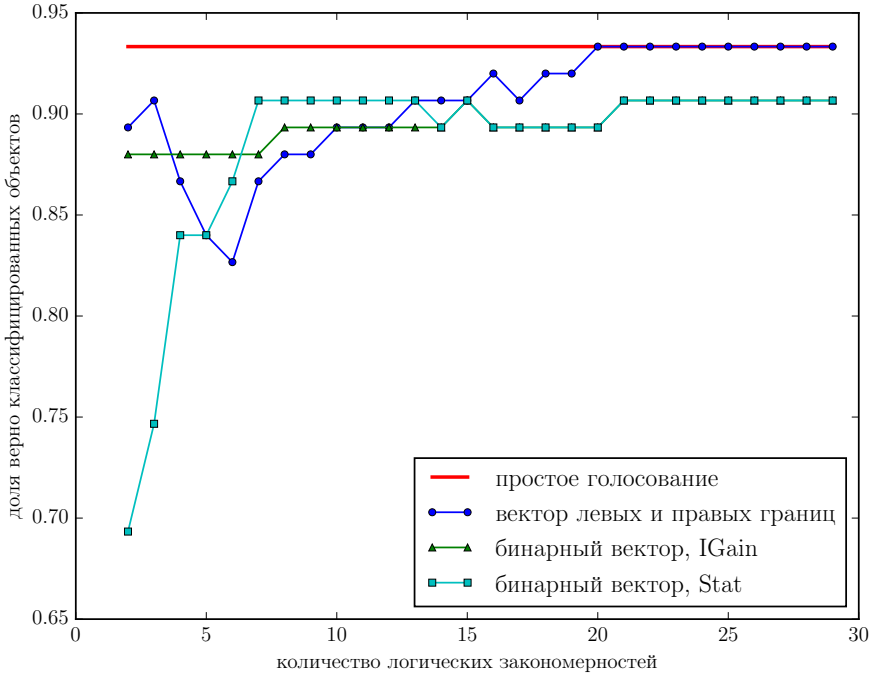
В формуле (4) используется обозначение $P_1, \dots, P_K$ — количество объектов в классе $1, \dots, K$ и $p_1, \dots, p_K$ — количество объектов класса, покрываемых закономерностью $\varphi(x)$. Наконец, $C_n^k = \frac{n!}{k!(n-k)!}$ обозначает биномиальный коэффициент.

4.3 Обсуждение и выводы

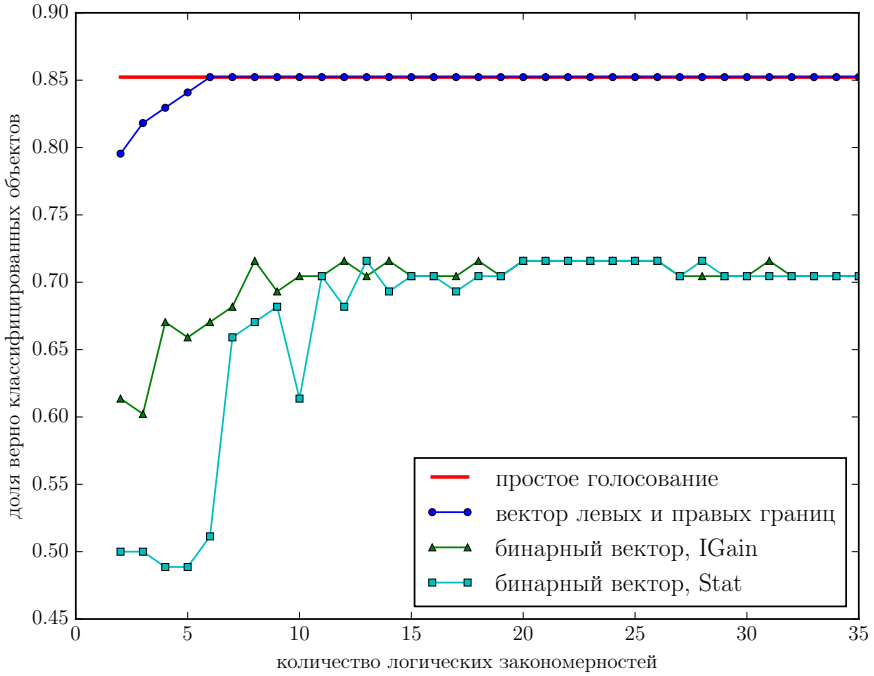
- Обработка логических закономерностей, использующая описание правил с помощью вектора левых и правых границ, получает результат, близкий к исходному набору логических закономерностей.

- Обработка логических закономерностей, использующая описание правил с помощью вектора левых и правых границ, восстанавливает логические закономерности, которые в целом показывают результаты лучше, чем результаты обработки с использованием бинаризованных описаний правил 1a, 1b, 2a, 2b.

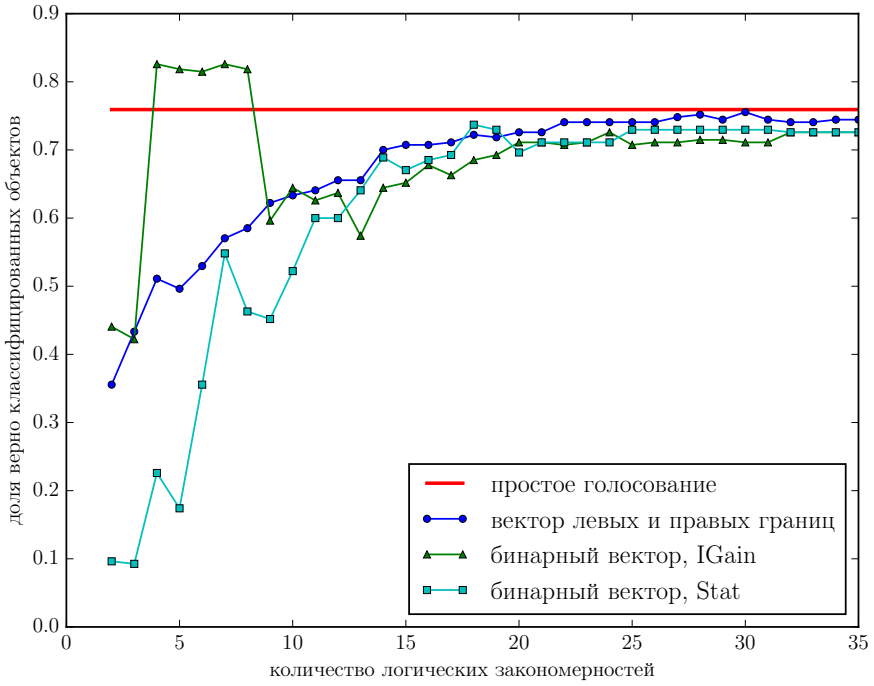
- Обработка с использованием бинаризованных представлений правил иногда работает лучше 3a. Если же исходно используются не чистые, а частичные закономерности, то представление с помощью вектора левых и правых границ более устойчиво 2b, 3b



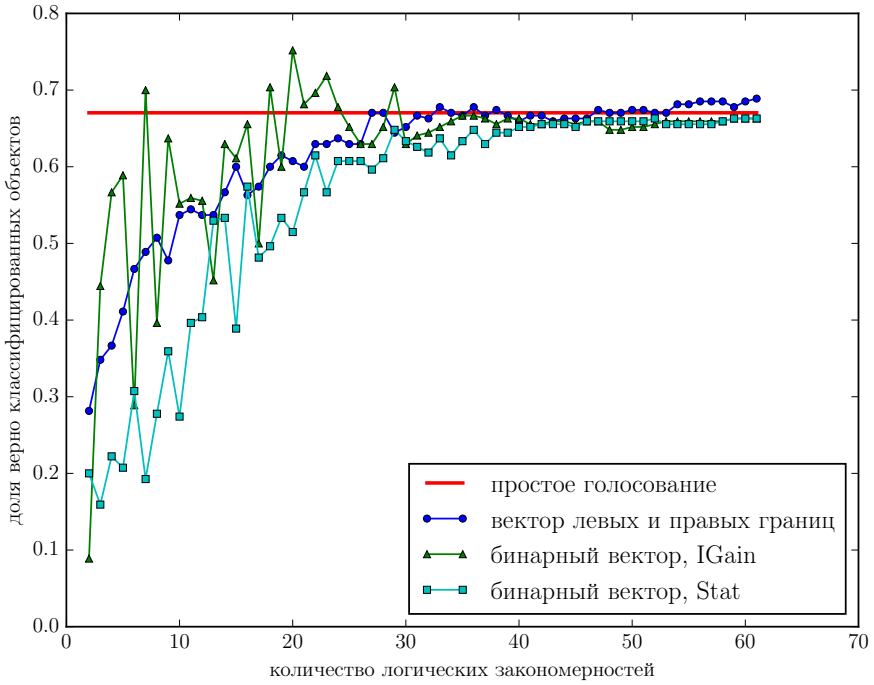
(a) Выборка Iris



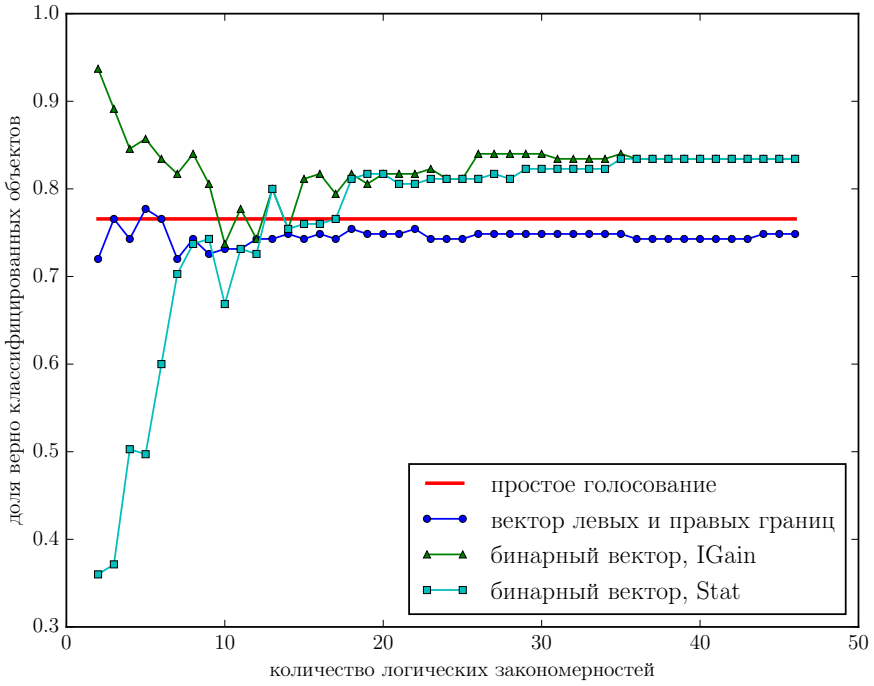
(b) Выборка Wine



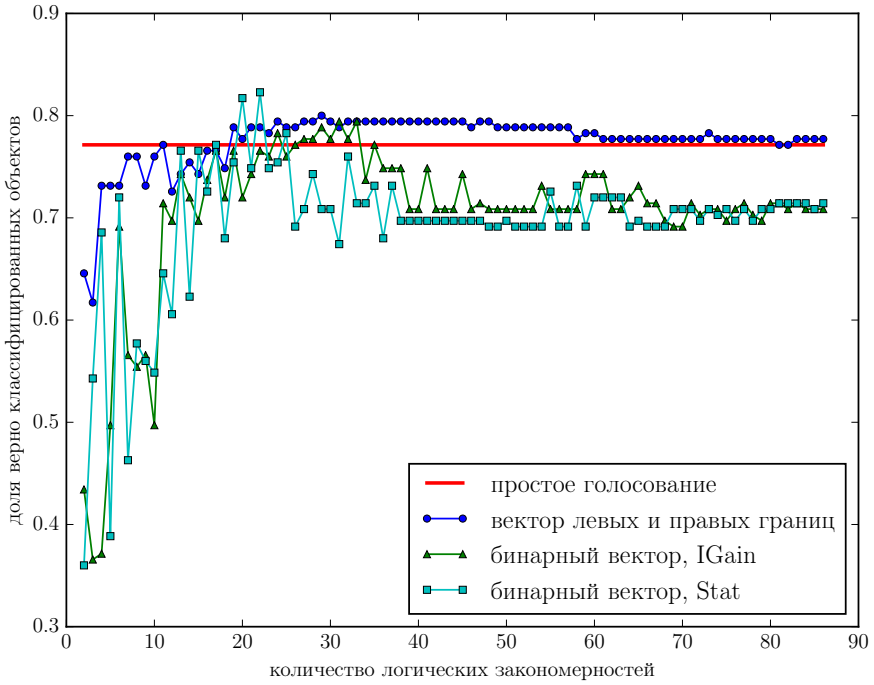
(a) Выборка Climate, исходные логические закономерности чистые (не покрывают объекты чужих классов)



(b) Выборка Climate, исходные логические закономерности частичные (среди покрываемых объектов не более 10% из чужих классов)



(a) Выборка Ionosphere, исходные логические закономерности чистые (не покрывают объекты чужих классов)



(b) Выборка Ionosphere, исходные логические закономерности частичные (среди покрываемых объектов не более 10% из чужих классов)

- В результате обработки множества логических закономерностей как с помощью бинаризованного представления, так и с помощью вектора левых и правых границ, можно получить сравнимое с исходным качество классификации при меньшем количестве использованных правил.

5 Заключение

В данной работе был предложен подход к обработке множеств логических закономерностей. В частности, в разделе 3 была сформулирована схема использования дисперсионного критерия, а в разделе 4 были поставлены эксперименты по исследованию этого подхода. С их помощью удалось сравнить использование бинаризованного описания логических закономерностей, предложенное в [11], и описание логических закономерностей с помощью вектора левых и правых границ, предложенное в данной работе. В результате предложенный в данной работе подход с использованием вектора левых и правых границ и взвешенной кластеризации в целом работает лучше, чем подход с бинаризованным описанием. Качество классификации сравнимо с исходным алгоритмом системы «Распознавание», но при этом размер множества логических закономерностей меньше.

A Система «Распознавание»

Система «Распознавание» [15] работает под управлением операционной системы семейства Microsoft Windows XP и выше. В этом приложении будет показано, как систему «Распознавание» можно использовать на UNIX-подобных операционных системах. Для этого нужно:

1. Установить свободное программное обеспечение Wine.
2. Настроить 32-битный префикс командой

   ```
   WINEARCH=win32 WINEPREFIX=~/.wine32 winecfg
   ```

3. Установить в созданный префикс недостающую библиотеку

   ```
   WINEPREFIX=~/.wine32 winetricks mfc42
   ```

4. Запустить систему «Распознавание» командой

   ```
   LC_CTYPE=ru_RU.utf8 WINEARCH=win32 WINEPREFIX=~/.wine32 \
   wine Recognition.exe
   ```

B Формат файлов *.tab

Система «Распознавание» [15] использует собственный формат файлов описания выборки. Здесь приведено описание этого формата:

Первая строка — заголовок. Представляет из себя числа, разделенные пробелом. Первое число — количество признаков каждого объекта выборки. Второе — количество классов. Далее идет число 0, за которым идет число объектов первого класса, затем сумма числа объектов первых двух классов, затем сумма числа объектов первых трех классов и так далее. Завершает заголовок число, обозначающее пропуск измерения в данных.

Каждая последующая строка содержит признаковое описание отдельного объекта, разные классы разделены пустой строкой.

Список литературы

Cohen W. W., *Singer Y.* A Simple, Fast and Effective Rule Learner // Proc. of the 16 National Conference on Artificial Intelligence. — 1999. — С. 335—342.

Quinlan J. R. Bagging, Boosting, and C4.5 // AAAI/IAAI, Vol. 1. — 1996. — С. 725—730.

Quinlan J. R. C4.5: Programs for machine learning. — Morgan Kaufmann, San Francisco, CA, 1993.

Cohen W. W. Fast Effective Rule Induction // Proc. of the 12th International Conference on Machine Learning, Tahoe City, CA. — Morgan Kaufmann, 1995. — С. 115—123.

Quinlan J. R. Induction of Decision Trees // Machine Learning. — 1986. — Т. 1, № 1. — С. 81—106.

Lloyd S. Least Squares Quantization in PCM // IEEE Trans. Inf. Theor. — Piscataway, NJ, USA, 2006. — Сент. — Т. 28, № 2. — С. 129—137.

Fürnkranz J. Pruning Algoruthms for Rule Learning // Machine Learning. — 1997. — № 27. — С. 139—172.

Macqueen J. B. Some methods for classification and analysis of multivariate observations // In 5-th Berkeley Symposium on Mathematical Statistics and Probability. — 1967. — С. 281—297.

Lichman M. UCI Machine Learning Repository. — 2013. — URL: `http://archive.ics.uci.edu/ml`.

Вайнцвайг М. Н. Алгоритм обучения распознаванию образов «Кора» // Алгоритмы обучения распознаванию образов / под ред. В. Н. Вапник. — М.: Советское радио, 1973. — С. 110—116.

Новиков М. С. Исследование задачи кластеризации логических закономерностей, представленных булевыми векторами. — 2014.

Воронцов К. В. Лекции по логическим алгоритмам классификации. — 2010. — URL: `http://www.machinelearning.ru/wiki/images/3/3e/Voron-ML-Logic.pdf`.

Рязанов В. В. Логические закономерности в задачах распознавания (параметрический подход) // Журнал вычислительной математики и математической физики. — 2007. — Т. 47. — С. 1793—1808.

Полякова М. П., *Вайнцвайг М. Н.* Об использовании метода «голосования» признаков в алгоритмах распознавания // Моделирование обучения и поведения. — М., 1975. — С. 25—28.

Журавлёв Ю. И., *Рязанов В. В.*, *Сенько О. В.* «Распознавание». Математические методы. Программная система. Практические применения. — М.: Фазис, 2006.