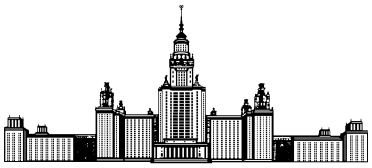
Московский государственный университет имени М. В. Ломоносова



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

Данный документ является образцом оформления дипломной работы для студентов кафедры Математических методов прогнозирования ВМК МГУ. Приведённые ниже рекомендации взяты из статьи «Написание отчётов и статей (рекомендации)» на вики-ресурсе `www.MachineLearning.ru`. Студенты, готовящие дипломную работу к защите, могут найти много полезной информации также в статьях «Научно-исследовательская работа (рекомендации)», «Подготовка презентаций (рекомендации)», «Защита выпускной квалификационной работы (рекомендации)» на том же ресурсе.

Аннотация обычно содержит краткое описание постановки задачи и полученных результатов, одним абзацем на 10–15 строк. Цель аннотации — обозначить в общих чертах, о чём работа, чтобы человек, совершенно не знакомый с данной работой, понял, интересна ли ему эта тема, и стоит ли читать дальше. Аннотация собирается в последнюю очередь путем легкой модификации наиболее важных и удачных фраз из введения и заключения.

1 Введение

Во введении рассказывается, где возникает данная задача, и почему её решение так важно. Вводится на неформальном уровне минимум терминов, необходимый для понимания постановки задачи. Приводится краткий анализ источников информации (литературный обзор): как эту задачу решали до сих пор, в чем недостаток этих решений, и что нового предлагает автор. Формулируются цели исследования. В конце введения даётся краткое содержание работы по разделам; при этом отмечается, какие подходы, методы, алгоритмы предлагаются автором впервые. При упоминании ключевых разделов кратко формулируются основные результаты и наиболее важные выводы.

Цель введения: дать достаточно полное представление о выполненном исследовании и полученных результатах, понятное широкому кругу специалистов. Большинство читателей прочтут именно введение и, быть может, заключение.

Во введении автор решает сложную оптимизационную проблему: как сообщить только самое важное, потратив минимум времени читателя, да так, чтобы максимум читателей поняли, о чём вообще идёт речь.

Введение лучше писать напоследок, так как в ходе работы обычно происходит переосмысление постановки задачи. Если же введение писать, когда работа еще не готова, задача усложняется вдвойне. В конце обычно приходит понимание, что всё получилось совсем не так, как планировалось в начале, и исходный вариант введения всё равно придётся переписывать. Кстати, к таким «потерям» надо относиться спокойно — в хорошей работе почти каждый абзац многократно переделывается до неузнаваемости.

Введение имеет много общего с текстом доклада на защите, поэтому имеет смысл готовить их одновременно.

1.1 Определения и обозначения

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. Рассмотрим эту группу алгоритмов подробнее, при этом будем придерживаться обозначений [10].

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 [10].

- 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* $[0,1]$

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. В. Рязановым в [11].

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

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

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 правил (rule induction)

К таким алгоритмам относятся «КОРА» [8], «IREP» и «RIPPER» [4], «SLIPPER» [1], «ID3» [5], «C4.5rules» [3], [2]. В качестве классического примера такого алгоритма можно привести двухклассовую версию «IREP».

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 объ-

екты, которые покрываются созданным правилом, исключаются из рассмотрения. Многие алгоритмы («КОРА», «ТЭМП» [10]) по смыслу очень похожи и отличаются только критериями останова и способами упрощения полученных правил.

2.2 Общая проблема алгоритмов поиска правил

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 [10], [12].

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 [6]. Проблема диверсификации, наконец, заключается в том, что довольно часто находятся очень похожие закономерности. Неформально объясним, почему это действительно плохо, на примере из [10]:

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, описывающих 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 нижних и верхних границ.

Представление с помощью бинарного вектора было исследовано в [9]. Идея заключается в том, чтобы каждой логической закономерности $\varphi(x)$ поставить в соответствие вектор $\boldsymbol{z} \in \{0,1\}^l$ так, что

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

Представление с помощью вектора верхних и нижних границ 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 $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n$ на $t \leq n$ непересекающихся классов $\boldsymbol{S} = \{S_1, S_2, \ldots, S_t\}$ таким образом, чтобы минимизировать следующий функционал:

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

Также для каждой закономерности можно учитывать «вес», то есть добавить константные коэффициенты β_j, с помощью которых выделяются информативные закономерности:

$$\boldsymbol{S}^* = \arg\min_{\boldsymbol{S}} \sum_{i=1}^{t} \sum_{x_j \in S_i} \beta_j \|x_j - \mu_i\|^2$$

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.

4 Вычислительные эксперименты

1. Обучающая выборка делилась пополам на обучение и контроль с сохранением доли каждого класса.

2. В системе «Распознавание» на обучающей выборке настраивался классификатор «логические закономерности»

3. Логические закономерности настроенного классификатора импортировались в написанную мною программу, в которой происходила их обработка и кластеризация с помощью дисперсионного критерия.

Выборка	Объекты	Классы	Признаки	Пропуски
Iris	150	50/50/50	4	нет
Wine	178	59/71/48	13	нет
Climate	540	46/494	11	нет
Ionosphere	351	126/255	34	нет

Таблица 1: Сводная таблица по использованным данным

4. По исходному набору логических закономерностей с помощью алгоритма «простого голосования» [10] проводилась классификация контрольной выборки.

5. Далее проводилась кластеризация исходного набора логических закономерностей на $2, 3, \ldots, n, \ldots$ кластеров.

6. Полученные на очередном шаге n центров кластеров использовались, как новые логические закономерности в алгоритме «простого голосования». Качество оценивалось по контрольной выборке.

7. В результате строился график зависимости качества классификации от числа кластеров.

4.1 Исходные данные и условия эксперимента

Исходные данные были взяты из репозитория UCI [7]. Ниже представлена сводная таблица по использованным данным 1

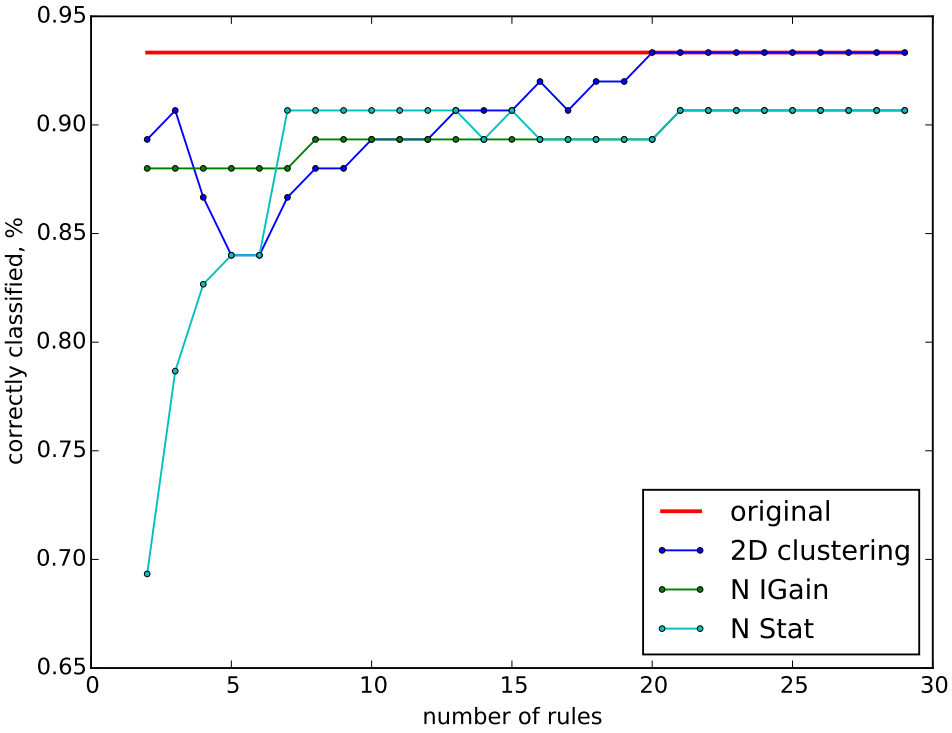
4.2 Результаты эксперимента

4.3 Обсуждение и выводы

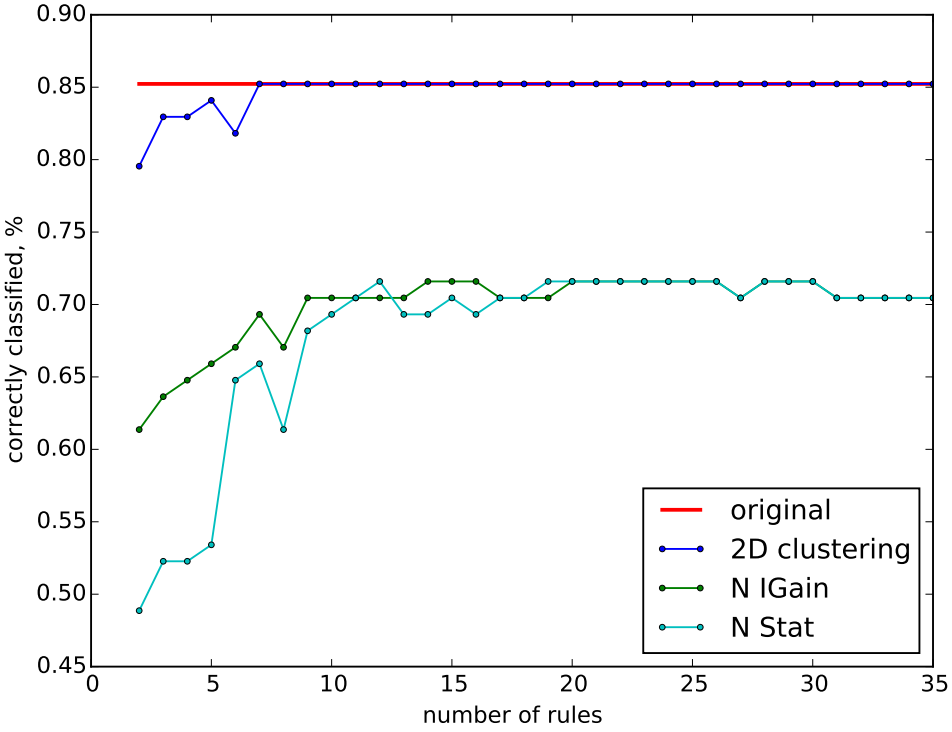
5 Заключение

В квалификационных работах последний раздел нужен для того, чтобы конспективно перечислить основные результаты, полученные лично автором.

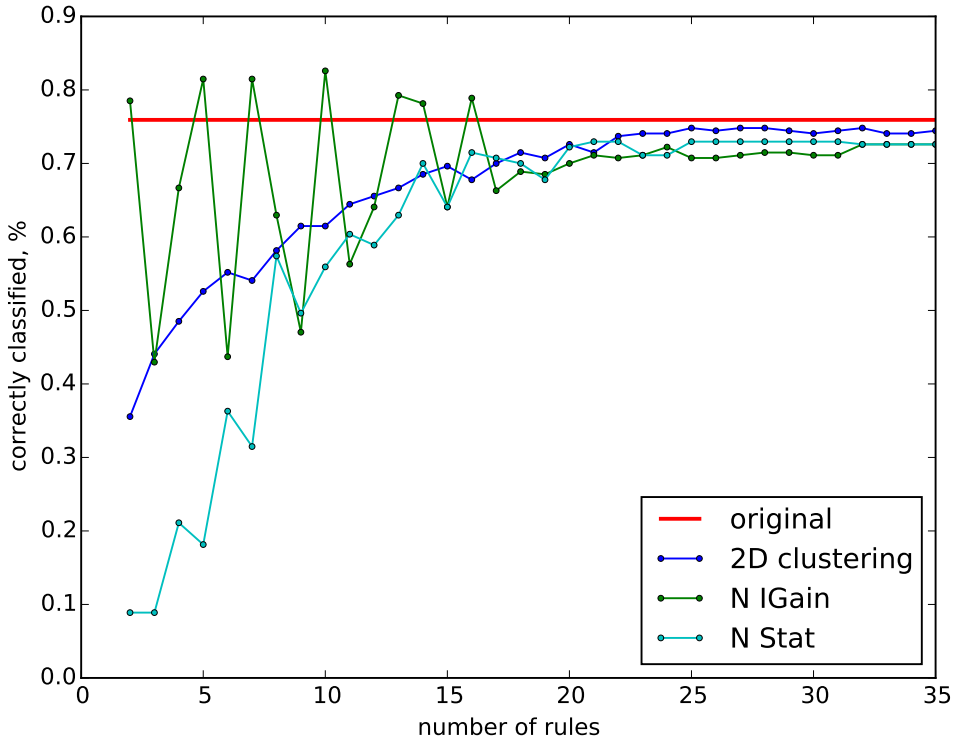
Результатами, в частности, являются:



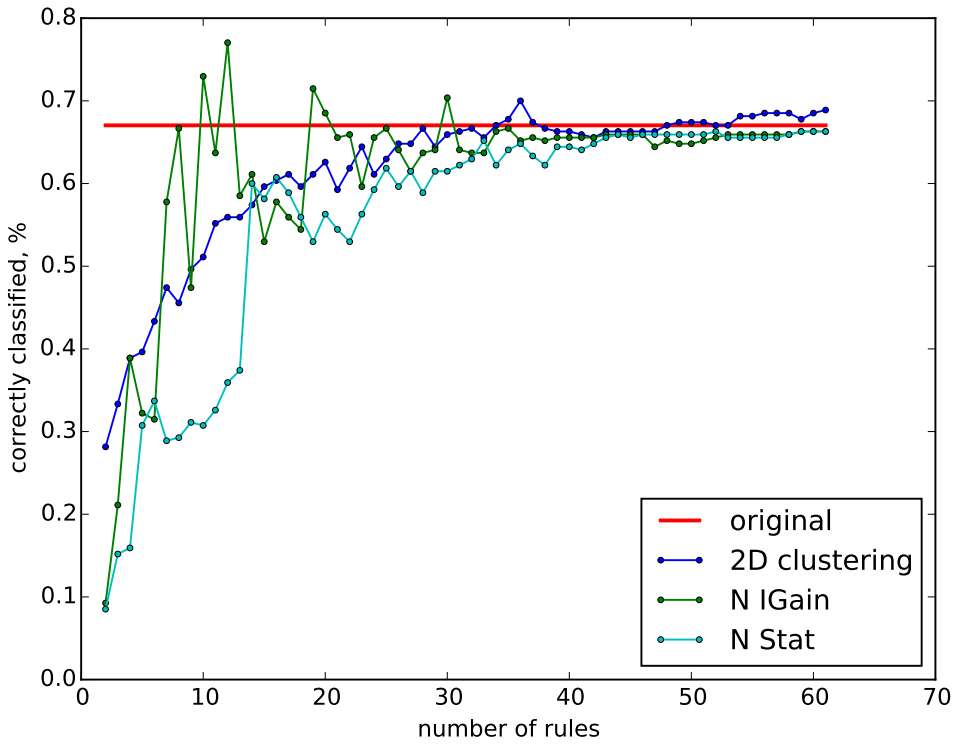
(a) iris results



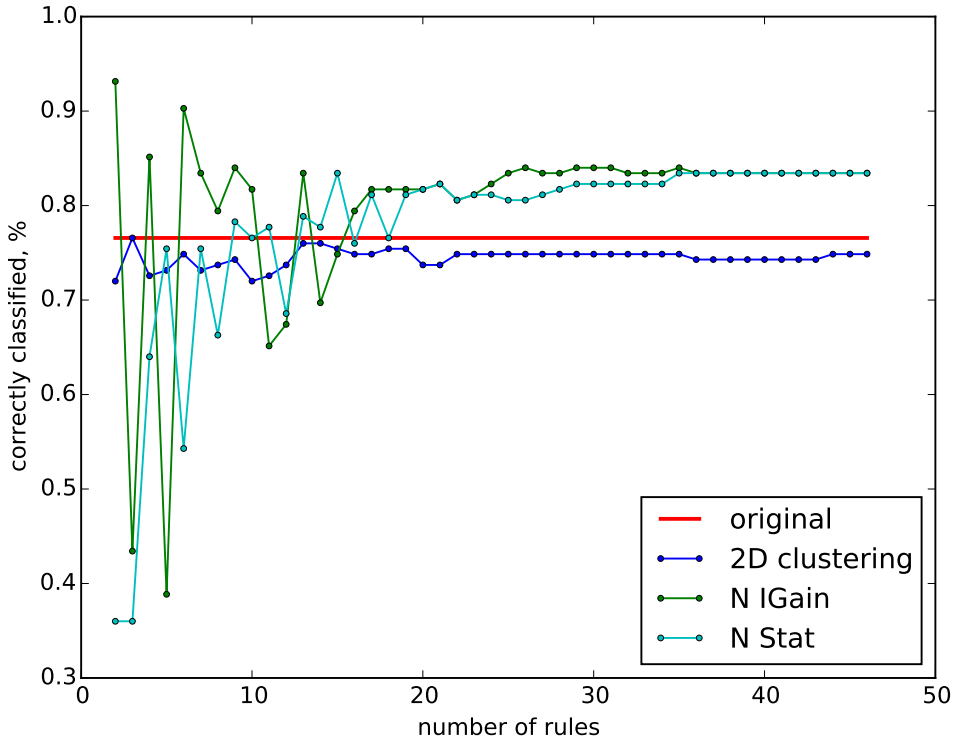
(b) wine results



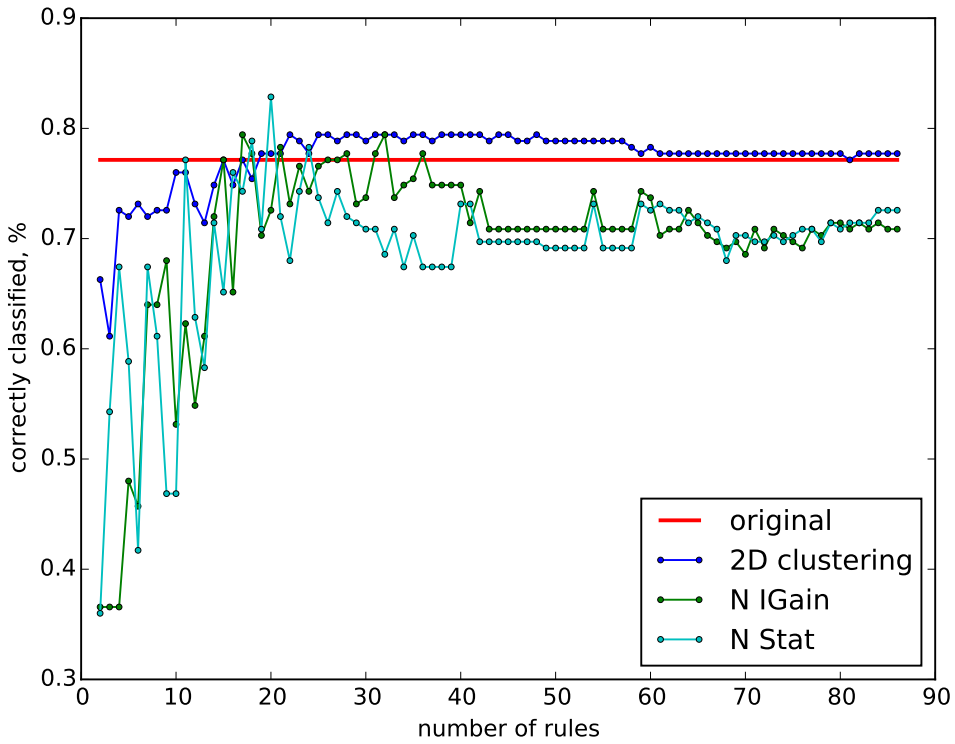
(a) Климат, только безошибочные логические закономерности



(b) Климат, каждая логическая закономерность ошибочно выделят до 10% объектов



(a) Ионосфера, только безошибочные логические закономерности



(b) Ионосфера, каждая логическая закономерность ошибочно выделят до 10% объектов

- Предложен новый подход к...
- Разработан новый метод..., позволяющий...
- Доказан ряд теорем, подтверждающих (опровергающих), что...
- Проведены вычислительные эксперименты..., которые подтвердили / опровергли / привели к новым постановкам задач.

Цель данного раздела: доказать квалификацию автора. Даже беглого взгляда на заключение должно быть достаточно, чтобы стало ясно: автору удалось решить актуальную, трудную, ранее не решённую задачу, предложенные автором решения обоснованы и проверены.

Иногда в Заключении приводится список направлений дальнейших исследований.

Список литературы необходим в любой научной публикации. В дипломной работе он обязателен. Дурным тоном считается: ссылаться на работы только одного-двух авторов (например, себя или шефа); ссылаться на слишком малое число работ; ссылаться только на очень старые работы; ссылаться на работы, которых автор ни разу не видел; ссылаться на работы, которые не упоминаются в тексте или которые не имеют отношения к данному тексту.

A Система «Распознавание»

Система «Распознавание» 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:

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

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

- Установить в созданный префикс недостающую библиотеку

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

- Запустить систему «Распознавание» командой

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

B Формат файлов *.tab

Система «Распознавание» 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.

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

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.