КОМПОЗИЦИЯ СЛУЧАЙНЫХ ИНКРЕМЕНТНЫХ ДЕРЕВЬЕВ И ВОССТАНОВЛЕНИЕ СТРУКТУРЫ ТАБЛИЦ[1]

П.Ю. Кудинов,
аспирант отдела вычислительных методов прогнозирования Вычислительного центра им. А.А. Дородницына Российской академии наук (ВЦ РАН)

В.А. Полежаев,
студент кафедры математических методов прогнозирования Московского государственного университета им. М.В. Ломоносова (ВМК МГУ)

Адрес: г. Москва, ул. Вавилова, д. 40
E-mail: pkudinov@gmail.com, valentin.polezhaev@gmail.com

В статье предлагается новый корректный инкрементный алгоритм классификации, основанный на построении композиции случайных деревьев (RIF). Рассмотрена задача распознавания структуры статистических таблиц, предложены постановки задач распознавания и признаки объектов. Проведено сравнение нового алгоритма с известным алгоритмом ITI. Алгоритм RIF показывает лучшие результаты на большинстве задач.

Ключевые слова: инкрементное машинное обучение, корректные алгоритмы, анализ текстов, информационный поиск, классификация, деревья решений, распознавание образов.

1. Введение

Во многих задачах распознавания есть потребность продолжать обучение на новых объектах обучающей выборки. Такой процесс называется динамическим обучением (on-line learning). После распознавания очередного объекта алгоритму передаётся метка класса этого объекта, которую он должен быть способен учесть, т. е. «дообучиться». Такого рода алгоритмы используются в полуавтоматических системах обработки информации. Главным преимуществом этих систем является повышение эффективности труда эксперта или оператора, т. к. с течением времени алгоритм совершает меньше ошибок на новых объектах. Во многих приложениях исправления оператора считаются абсолютно верными, поэтому к этим алгоритмам предъявляется *требование корректности*

[1] Работа выполнена при финансовой поддержке РФФИ, проекты №11-07-00480-а и 10-07-00609-а.

[1, 2], т. е. гарантия безошибочной классификации объектов обучающей выборки.

В настоящее время актуальной является задача создания системы информационного поиска по социально-демографической, экономической, финансовой статистической информации, свободно доступной в интернете в виде текстовых таблиц. Основной функцией системы должен быть вывод агрегированных статистических таблиц по запросам пользователей. Первым шагом на пути реализации такой системы является разработка методов извлечения статистических показателей из таблиц [3, 4]. Возникающие при этом задачи распознавания предлагается решать методами динамического обучения. Основной сложностью является большой объём выборки, который будет достигать десятков тысяч таблиц в год. Поэтому к динамическим алгоритмам предъявляется не только требование корректности и малый процент ошибок, но и высокая вычислительная эффективность. В работе [4] авторами исследовались инкрементные алгоритмы для задач классификации, возникающих при извлечении данных из статистических таблиц. В результате проведённых экспериментов оказалось, что наиболее эффективным алгоритмом является Incremental Tree Induction (ITI, [5, 6]), который позволяет достичь приемлемого качества классификации и удовлетворяет требованию корректности. Основным недостатком данного алгоритма является необходимость постоянного перестроения и упрощения дерева, без выполнения которых происходит сильное его переобучение. Выполнение этих процедур занимает основное время работы алгоритма. В настоящей работе описываются основные свойства алгоритма ITI и предлагается новый корректный инкрементный алгоритм, который строит композицию случайных инкрементных деревьев. Показаны преимущества этого алгоритма перед ITI и рассматривается модификация с отбором деревьев.

2. Решающие деревья

Обозначим всё множество объектов через X, а множество признаков – через $\mathsf{F}=\{f_1, \ldots, f_M\}$, где $f_i : X \to D_{f_i}$. Если $|D_{f_i}|<\infty$, то f_i – номинальный признак, иначе – числовой. Множество классов обозначим через Y. Пусть дана обучающая выборка $X^l=\{(x_i, y_i)_{i=1}^l\}$, l пар объект-ответ. Решающее дерево – алгоритм классификации $X \to Y$, который представим в виде множества вершин $T=\{v_i\}$, корень дерева v_0. Каждой узловой вершине v соответствует (L_v, R_v, β_v), где L_v, R_v – ссылки на левое и правое поддерево, а β_v является логическим условием. Будем рассматривать $\beta_v(x)=\mathrm{I}[f_j(x)<a]$ в случае числового признака, и $\beta_v(x)=\mathrm{I}[f_j(x)=a]$ в случае номинального. Каждой вершине v соответствует метка класса. Классификация осуществляется путём последовательной проверки условий в узлах дерева начиная от корня дерева. Если $\beta_v(x)=0$ то $v=L_v$, иначе – $v=R_v$. Обучение алгоритма состоит в рекурсивном разбиении выборки на две части по наиболее информативному логическому условию (**Алгоритм 1**). На каждом этапе выбор наилучшего условия осуществляется перебором всевозможных предикатов, то есть перебором всевозможных разбиений выборки и вычислением информативности каждого из них. Информативность может быть вычислена с помощью энтропийного критерия информативности, индекса Джини или других эвристических критериев [7].

Алгоритм 1. Обучение дерева (TrainTree)
Вход: $X^l=\{(x_i, y_i)_{i=1}^l\}$ – множество пар объект-ответ;
Выход: v – корень дерева;
если в X^l представлены объекты как минимум 2 классов **то**
наилучшего β= предикат, разделяющий X^l
L:=$TrainTree(\{x \in X^l \mid \beta(x)=0\})$; (обучаем левое поддерево)
R:=$TrainTree(\{x \in X^l \mid \beta(x)=1\})$; (обучаем правое поддерево)
вернуть $v=\langle L, R, \beta\rangle$;
иначе
вернуть $v=\langle y\rangle$, y – класс объектов, представленных в X^l;

Пусть имеется подвыборка

$$A=\{(a_i, b_i)\} \subseteq X^l,\ 1,, i,, |A|$$

и её разбиение на два непересекающихся подмножества A_1, A_2, $A_1 \cap A_2=\varnothing$. Обозначим через

$$p_y(A)=\frac{1}{|A|}\sum_{i=1}^{|A|}[b_i=y]$$

относительное число объектов класса y.

Икрементное построение дерева решений

Алгоритм Incremental Tree Induction (ITI, [5, 6]) предполагает инкрементное построение бинарного дерева решений. Дерево решений алгоритма ITI отличается от дерева решений ID3 тем, что в узловой вершине v помимо тройки (L_v, R_v, β_v) хранится пятёрка $(L_v, R_v, \beta_v, X_v, s_v)$, где L_v, R_v, β_v, как и в обычном дереве решений, левая, правая вершина и предикат соответственно, $X_v \subset X^l$ – список объектов, прошедших через узел v, s_v – состояние узла. Если через него прошли новые объекты, то s_v=1, иначе s_v=0. Перед началом инкрементного обучения производится обучение на части выборки $X_0 \subset X^l$. Далее начинается этап инкрементного обучения. После классификации очередного объекта x происходит его встраивание в дерево (**Алгоритм 2**). Начиная от корневого узла происходит последовательный переход к дочерним узлам в соответствии с предикатами, пока очередной узел не окажется листом. В каждом посещённом узле v устанавливается состояние s_v=1 и к списку объектов добавляется новый объект x. В случае если класс нового объекта совпадает с классом, установленным в листе, то он добавляется к списку объектов. Иначе происходит поиск наилучшего разбиения объектов, хранимых в листе, и он превращается в узел, для которого строятся потомки. Таким образом, в корне дерева хранится информация обо всей выборке; на следующем уровне в каждом узле или листе хранится информация, которая в совокупности даёт информацию обо всей выборке, и т. д.

Алгоритм 2. Добавление объекта в дерево (AddToTree)
Вход: T, x, y – обученное дерево, объект для классификации, правильный ответ;
Выход: T,– обновленное дерево;
  $v:=v_0$;
  пока v – узловая вершина
    $X_v:= X_v \cup x$;
    s_v=1;
    если $\beta_v(x)$=0, **то**
      $v:= L_v$; (переход влево)
    иначе
      $v:= R_v$; (переход вправо)
  если y=c_v, то
    $X_v:= X_v \cup x$;
  иначе
    $Z:= X_v \cup x$; (список объектов в узле)
    β_v = наилучший предикат для Z;
    L_v:=$TrainTree(\{z \in Z | \beta(z)=0\})$; (обучаем левое поддерево)
    R_v:=$(\{z \in Z | \beta(z)=1\})$; (обучаем правое поддерево)
    $v := \langle L, R, \beta, X_v, s_v \rangle$;
  вернуть T.

Для обеспечения наилучшего качества при добавлении нового объекта можно было бы перестраивать дерево заново, однако это вычислительно не эффективно. В [6] описывается способ инкрементного преобразования дерева, который заключается в проверке того, что во всех узлах установлены предикаты, имеющие наилучшую информативность. Если это не так, то происходит поиск нового условия преобразования дерева, называемое трансформацией. Однако поиск лучшего предиката имеет высокую вычислительную сложность – его выполнение занимает большую часть времени работы алгоритма.

3. Алгоритм RIF

Основные затраты при построении инкрементного дерева и выполнении транспозиции приходятся на поиск наилучшего предиката, то есть перебора разбиений. Одним из способов существенно сократить эти затраты является сокращение числа признаков, используемых при переборе.

Алгоритм Random Forest [8] представляет собой композицию решающих деревьев. Он основывается на двух принципах: бэггинге и методе случайных подпространств. Применительно к решающим деревьям это означает, что каждое новое дерево строится по новой случайной выборке с повторениями (полученной из исходной) и по случайному поднабору признаков. В работе предлагается новый алгоритм (**Алгоритмы 3, 4**) построения случайного инкрементного леса деревьев (Random Incremental Forest, RIF), результатом работы которого является композиция случайных инкрементных деревьев Random Incremental Tree (RIT). RIF можно представить в виде множества пар «дерево, набор признаков», RIF=$\{\langle RITree_i, M_i \rangle\}_{i=1}^{p}$, где $RITree_i$ – инкрементное дерево, построенное по случайному набору признаков $M_i \subseteq$ F.

Алгоритм 3. Обучение композиции (RIF.Train)
Вход: $X^l = \{(x_i, y_i)_{i=1}^l\}, K, M$ – множество пар объект-ответ, число деревьев, число используемых признаков соответственно;
Выход: $RIF = \{\langle RITree_i, M_i \rangle\}_{i=1}^K$ – множество пар дерево-признаковое подпространство;
  для i=1,..., K
    M_i:= случайное подмножество признаков из M элементов; $RITree_i := TrainTree(X^l, M_i)$;
  вернуть $\{\langle RITree_i, M_i \rangle\}_{i=1}^K$;.

3.1. Обучение

По начальной обучающей выборке строится композиция решающих деревьев по аналогии с Random Forest. Для построения каждого дерева используется одна и та же исходная выборка без перемешивания с повторениями, от принципа бэггинга приходится отказаться для обеспечения корректности композиции.

Для каждого дерева случайно генерируется поднабор из M_i=|F| признаков. Каждое дерево строится по стандартному алгоритму построения дерева решений с небольшим изменением: при поиске наилучшего разбиения выбирается случайный признак из поднабора признаков.

3.2. Классификация и внедрение нового объекта

Объект даётся на классификацию всем деревьям в композиции и относится к тому классу, за который проголосовало большее число деревьев. Внедрение нового объекта происходит во все деревья композиции (**Алгоритм 4**).

Алгоритм 4. Дообучение композиции новым объектом (RIF.Update)

Вход: $RIF=\{\langle RITree_i, E\rangle\}_{i=1}^{K}$, x, y — обученная композиция, новый объект и правильный ответ соответственно;

Выход: $RIF = \{\langle RITree_i, M_i\rangle\}_{i=1}^{K}$ — обновлённая композиция;

для i=1,..., K

$RITree_i := AddToTree(RITree_i, x, y)$; — обновление каждого дерева, простое внедрение объекта;

вернуть RIF.

4. Отбор деревьев

Так как набор признаков в каждом дереве случаен, получающиеся в ходе инкрементного обучения деревья могут сильно отличаться по качеству. В случае длинных обучающих выборок можно осуществить отбор деревьев по некоторому критерию качества. Дополним каждое дерево композиции двумя показателями: τ — число объектов, которые были классифицированы деревом, и μ — число ошибок дерева. Эти показатели обновляются после классификации и внедрения каждого нового объекта.

4.1. Процедура отбора деревьев

Введём несколько параметров. K_0 — начальное число деревьев, стоит выбирать разумно большим, в зависимости от темпа отбора и размеров задачи. W — число худших деревьев, которые будут удалены. B — число лучших деревьев, на основе которых будут строиться новые деревья. При построении нового дерева признаки случайно выбираются из поднабора признаков, на которых построены лучшие деревья. Будем осуществлять взвешивание признаков — если некоторый признак оказался в признаковых наборах нескольких деревьев, то его вес и соответственно вероятность его выбора в набор для нового дерева увеличивается. ΔK– число новых деревьев, которые будут построены на основе признакового подпространства лучших деревьев.

Процесс отбора деревьев представляет собой последовательность следующих операций (**Алгоритм 5**) и выполняется с заданным периодом. Значение W–ΔK характеризует темп отбора, значение B характеризует «ширину» отбора. Чем меньше B, тем меньше набор признаков для выбора при построении новых деревьев. Также имеет смысл определить минимальное «время жизни» τ_0 для деревьев и минимальное число деревьев K_{min}. Слишком «молодые» деревья не участвуют в процедуре отбора, так как их статистика малоинформативна, также процедура отбора не выполняется, если число деревьев достигло минимума, то есть существует риск потери в качестве классификации.

Алгоритм 5. RIFTS.SelectTrees

Вход: $RIFTS = \{\langle RITree_i, M_i\rangle\}_{i=1}^{K}, X^l, W, B, \Delta K, K_{min}, \tau_0$ — обученная композиция, вся доступная обучающая выборка, параметры отбора деревьев соответственно;

Выход: $RIFTS = \{\langle RITree_i, M_i\rangle\}_{i=1}^{M}$ — обновлённая композиция;

для всех деревьев с номерами i=1,..., $K : \tau\ \tau_0$

вычислить качество q_i;

Отсортировать деревья по убыванию q_i, i=1,..., K;

Удалить min (W, $K - K_{min}$) последних деревьев;

$A = \bigcup_{i=1,\dots,B} M_i$; — признаки лучших деревьев;

для j = 1,..., ΔK

M_j = случайные M признаков из A;

RIT_j = TrainTree(X^l, M_j);

$RIFTS.Add(RIT_j)$; — добавление нового дерева в композицию;

вернуть $RIFTS$.

4.2. Оценка качества дерева

Так как деревья строятся в разные моменты и не ошибаются на своих обучающих выборках, то для их сравнения недостаточно учитывать только величину μ (очевидно, построенные позже деревья всегда будут иметь значение μ меньше), также нужно учитывать τ. Деревья можно сравнивать по относительному числу ошибок (величина μ/τ). Рассмотрим два дерева, которые имеют одинаковую относительную ошибку μ, но одно из деревьев имеет значительно большую величину τ. Такое дерево кажется более надежным, но это никак не отражается в таком критерии. Можно выводить различные эвристические формулы с участием величин τ и μ, но адекватность критериев будет, как правило, зависеть от размеров задачи.

Допустим, что классификация объекта – случайная величина η из распределения Бернулли с параметром p. Пусть имеется n испытаний и m ошибок. Тогда по теореме Муавра-Лапласа можно получить верхнюю оценку вероятности ошибки:

$$\overline{p} = \frac{m}{n} + \Phi_{\frac{1-\alpha}{2}} \sqrt{\frac{m(n-m)}{n^3}} \quad .$$

Полученное выражение для $\overline{p}$ используется в качестве критерия качества дерева (при $n=\tau$, $m=\mu$).

5. Восстановление структуры статистических таблиц

Составители статистических таблиц используют различные средства для разделения таблицы на ячейки и блоки и различные типы логической структуры таблиц. Для её распознавания используются методы динамического обучения. Рассмотрим квадратную сетку $G^{M\times N}$ из M строк и N столбцов и зададим её полное покрытие непересекающимися прямоугольными областями. Элемент покрытия будем называть *ячейкой*. Множество всех ячеек обозначим через C. Каждой ячейке $c \in C$ поставим в соответствие текстовую строку $text(c)$, возможно пустую, которую назовём *содержимым ячейки* или её *текстом*. Будем полагать, что каждая ячейка c имеет тип $type(c) \in \{data, key, empty\}$. *Ячейка данных* (*data*) содержит ровно одно числовое значение. *Ячейка описания* (*key*) содержит текстовую строку, состоящую из словесных описаний одного или нескольких ключей. *Пустая ячейка* (*empty*) не содержит значимой информации, её содержимое игнорируется. *Статистической таблицей T* будем называть четвёрку $\langle G^{M\times N}, C_V, C_K, R\rangle$, где C_V – множество ячеек данных, C_K – множество ячеек описаний, отображение $R: C_V \to 2^{C_K}$ ставит в соответствие каждой ячейке данных множество ячеек описания, т. е. задаёт структуру таблицы.

5.1. Логическая структура таблиц

Отображение R определяется взаимным расположением ячеек данных и описаний и стилевым оформлением ячеек. Для таблицы *простой структуры* (*табл. 1*) оно определяется следующим образом: для каждой ячейки данных выбираются все ячейки описания, пересекающиеся со строкой или столбцом рассматриваемой ячейки.

Первым шагом к построению отображения R яв-

Таблица 1.

Статистическая таблица простой структуры

	2003	2004	2005	2006	2007	2008	2009
Всего в экономике	107,0	106,5	105,5	107,5	107,5	104,8	95,8
Сельское хозяйство, охота и лесное хозяйство	105,6	102,9	101,8	104,3	105,0	110,7	105,0
Рыболовство, рыбоводство	102,1	104,3	98,5	101,6	103,2	95,5	109,2
Добыча полезных ископаемых	109,2	107,3	106,3	103,3	103.1	101,0	96,1
Обрабатывающие производства	108,8	109,8	106,0	108,5	108,4	102,6	96,1
Производство и распределение электроэнергии, газа и воды	103,7	100,7	103.7	101,9	97,5	102,1	96,3
Строительство	105,3	106,8	105.9	115,8	112,8	109,1	91,4
Оптовая и розничная торговля, ремонт автотранспортных средств, мотоциклов, бытовых изделий и предметов личного пользования	109,8	110,5	105,1	110.8	104.8	108,1	92,1
Гостиницы и рестораны	100,3	103.1	108,5	109,2	108,0	109,2	87,1
Транспорт и связь	107,5	108,7	102,1	110,7	107,5	106.5	100,1
Операции с недвижимым имуществом, аренда и предоставление услуг	102,5	101,3	112,4	108,2	117,1	107,9	98,7

ляется классификация типов ячеек, т. е. построение множеств C_V и C_K. Положение ячейки $c \in C$ в таблице описывается координатами левого верхнего $(r_1(c), c_1(c))$ и правого нижнего $(r_2(c), c_2(c))$ углов прямоугольника по сетке $G^{M \times N}$. Для каждой ячейки $c \in C$ генерируются следующие признаки:

1. $f_1(c)$ – количество чисел;
2. $f_2(c)$ – количество слов;
3. $f_3(c)$ – количество символов;
4. $f_4(c) = (r_1(c) + r_2(c)) / 2M$ – вертикальное положение;
5. $f_5(c) = (c_1(c) + c_2(c)) / 2N$ – горизонтальное положение;
6. $f_6(c) = r_2(c) - r_1(c)$ – число вертикально объединённых элементов сетки;
7. $f_7(c) = c_2(c) - c_1(c)$ – число горизонтально объединённых элементов сетки.

5.2 Таблица с суперстроками

Таблица может быть разделена на несколько частей в соответствии со значениями некоторого показателя. Например, данные по мужской и женской занятости, данные за разные годы, абсолютные и относительные данные одних и тех же показателей. Для совмещения таких данных в одной таблице составители часто используют суперстроки (*табл. 2*).

Таблица 2.

Статистическая таблица с суперстроками

	Общая численность безработных			Численность официально зарегистрированных безработных		
		в % к			в % к	
	тыс. человек	соответствующему периоду предыдущего года	предыдущему периоду	тыс. человек	соответствующему периоду предыдущего года	предыдущему периоду
2010 г.						
Январь	6832	105,1	110,7	2202	129,0	102,6
I квартал (в среднем за месяц)	6555	96,2	108,0	2244	114,2	108,5
2011 г.						
Январь	5815	85,1	107,8	1609	73,1	101,2

Рассмотрим задачу классификации строк таблицы на два класса: «обычная строка» и «суперстрока». Для каждой строки будем строить следующий набор признаков:

1. $f_1(c)$ – количество ячеек в строке;
2. $f_2(c) = N$ – ширина таблицы;
3. $f_3(c)$ – высота строки;
4. $f_4(c)$ – количество пустых ячеек;
5. $f_5(c) = (c_1(c) + c_2(c)) / 2N$ – количество не пустых ячеек.

5.3 Вложенные ячейки

Ещё одним часто встречающимся приёмом оформления таблиц является использование вложенных ячеек, когда несколько ячеек сдвигаются на один уровень вправо (*табл. 3*). Для определения вложенности решается задача классификации, в которой объектами являются пары последовательно идущих ячеек $p = (x_1, x_2)$ в левом блоке ячеек описания. Выделяются три класса: «ячейки находятся на одном уровне», «x_2 сдвинута вправо относительно x_1» и «x_2 сдвинута влево относительно x_1». Для этих объектов вычисляется следующий набор признаков:

1. $f_1(p)$ – текст x_1 заканчивается на «:»;
2. $f_2(p)$ – количество начальных пробельных символов в тексте x_2;
3. $f_3(p)$ – тип первого непробельного символа в x_2: «цифра», «буква» или «знак»;
4. $f_4(p)$ – первая буква в x_1 является прописной;
5. $f_5(p)$ – первая буква в x_2 является прописной;
6. $f_6(p) = r_2(x_1) - r_1(x_1)$ – высота x_1;
7. $f_7(p) = c_2(x_2) - c_1(x_2)$ – высота x_2.

Таблица 3.

Фрагмент таблицы с вложенными ячейками

	2002 г.			
	тыс. человек		распределение по полу, %	
	жен.	муж.	жен.	муж.
Все население	77562	67605	53	47
в том числе в возрасте, лет:				
0-9	6515	6825	49	51
10-19	11390	11817	49	51
20-29	10982	11097	50	50
30-39	10113	9939	50	50
40-49	12575	11578	52	48

6. Эксперименты

Для оценки качества алгоритма RIF использовались задачи из репозитория UCI [9] и реальные выборки статистических таблиц. Ошибка вычислялась как средняя величина по 20 повторениям с перемешиванием обучающей выборки. В качестве критерия информативности разбиения использовался индекс Джини.

6.1. Оптимальное число деревьев

Рассмотрим некоторые задачи, посмотрим зависимость качества от числа деревьев и сравним с результатами, полученными на этих же задачах, алгоритма ITI с транспозицией и без. Период транспозиции выбирался равным 50 или 100 в зависимости от размеров задачи.

В *табл. 4* приведены найденные наилучшие пары ошибка-число деревьев для каждой задачи. Эксперименты проводились для значений числа деревьев начиная с 10 и до 250 с шагом 5. На *рис. 1* приведён график зависимости процента ошибок от числа деревьев для задачи Heart.

Таблица 4.

Оптимальное число деревьев и ошибка композиции (в процентах)

Задача	Число деревьев RIF	Ошибка RIF	Ошибка ITI с транспозицией	Ошибка ITI без транспозиции
Ionosphere	85	7.6	12.8	12.6
Car	130	9.3	7.3	14.5
German	100	27.4	34.2	34.8
Heart	75	21.3	27.9	30.1
Yeast	115	47	52.1	55.3

Почти на всех задачах заметен выигрыш в качестве алгоритма RIF по отношению к ITI, на задачах Ionosphere и WBC алгоритм RIF работает значительно лучше.

6.2. Отбор деревьев

Эксперименты проводились только на достаточно больших задачах. Композиция без отбора RIF состояла из 50 деревьев.

Результаты эксперимента описаны в *табл. 5*. Видно, что отбор деревьев даёт улучшения в работе алгоритма не всегда, более того на задачу Pen отбор оказал отрицательное влияние.

Таблица 5.

Результаты отбора деревьев

Задача	Параметры отбора RIFTS (K_0, B, W, ΔK, K_{min})	Ошибка RIFTS, %	Ошибка RIF (без отора), %
Car	(160,10,10,7,50)	8.1	10.2
Nursery	(200,8,10,7,50)	4.3	6.4
Optical	(160,8,10,7,50)	13.2	12
Pen	(160,8,10,7,50)	9.9	4.1

6.3. Задачи распознавания структуры таблиц

Алгоритмы RIF и ITI сравнивались на коллекции из 1000 таблиц. Начальная обучающая выборка составляла 50 объектов, после чего запускалось динамическое обучение. Все эксперименты проводились 20 раз со случайно перемешанной выборкой. Композиция RIF состояла из 50 деревьев. График зависимости доли ошибок от числа обучающих объектов для задачи CT представлен на *рис. 1*, доверительные интервалы приведены в *табл. 6*.

Таблица 6.

Доверительные интервалы частоты ошибок алгоритмов (в процентах) на последнем объекте

	ITI			RIF		
	CT	SR	CI	CT	SR	CI
Мин.	0,05	0,02	0,6	0,04	0,02	1,3
Сред.	0,08	0,04	1,7	0,07	0,03	2,8
Макс.	0,13	0,08	2,5	0,11	0,05	4,4

Задача распознавания типа ячеек (CT) представлена 28624 объектами, 8 числовыми признаками. На этой задаче ITI использовался без транспозиции из-за больших вычислительных затрат. Оба алгоритма справились с задачей достаточно успешно, средняя ошибка составила менее 0,1%. Следует заметить, что доверительный интервал ошибки у RIF меньше, чем у ITI.

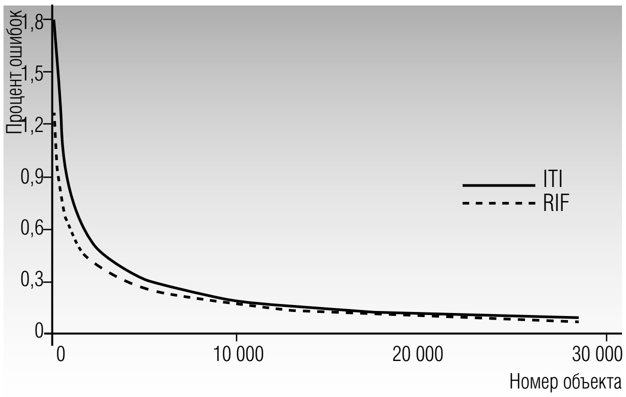

Рис. 1. Задача распознавания типа ячеек (CT).

Задача распознавания суперстрок (SR) представлена 102217 объектами, 5 числовыми признаками. На ней алгоритм ITI также использовался без транспозиции, для этой задачи справедливы выводы, аналогичные выводам по предыдущей задаче. Ошибка составила менее 0,1%.

Задача распознавания вложенных ячеек (CI) представлена 370 объектами, 4 номинальными и 7 числовыми признаками. Относительно небольшая длина выборки позволила использовать ITI с транспозицией (с периодом 50), в результате ITI справился с задачей лучше – 1,7% ошибок против 2,8% ошибок у RIF. Наличие транспозиции оказалось существенным для этой задачи.

6.4. Выводы

1. Композиция случайных деревьев RIF работает лучше, чем ITI, на некоторых задачах качество возрастает в 2 раза, но также есть задачи, где ITI незначительно опережает RIF.

2. Для таких задач, очевидно, критично наличие транспозиции (или перестроения дерева), то есть этот проигрыш можно устранить добавив какую-нибудь стратегию перестроения деревьев в композиции, например включение отбора деревьев, причем строить и удалять одинаковое число деревьев.

3. Композиция случайных деревьев RIF работает значительно быстрее, чем ITI с малыми периодами транспозиции и/или на больших задачах.

4. Композиция случайных деревьев с отбором деревьев RIFTS позволяет улучшить качество.

7. Заключение

В статье описаны особенности и недостатки известного инкрементного алгоритма ITI. Предложен новый корректный инкрементный алгоритм RIF, основанный на построении композиции случайных инкрементных деревьев и предложена модификация алгоритма с отбором деревьев (RIFTS). Проведённые вычислительные эксперименты на задачах репозитория UCI показали улучшение качества работы алгоритма на большинстве задач. На некоторых задачах прирост качества весьма значителен, не тратится много времени на операцию транспозиции. Отбор деревьев также, как правило, даёт прирост в качестве, причём величина прироста зависит от времени работы алгоритма – чем продолжительнее отбор, тем больше прирост качества. Рассмотрена задача распознавания структуры статистических таблиц, необходимой для построения поисковой системы статистической информации. Предложены постановки трёх задач распознавания структуры, перечислены признаки объектов, учитывающие специфику статистических таблиц. Эксперименты на реальной выборке показали качество классификации на уровне 96-99% для разных задач.

8. Благодарности

Авторы выражают благодарность своему научному руководителю профессору НИУ ВШЭ, д.ф.-м.н. К.В. Воронцову, а также профессору НИУ ВШЭ, д.т.н. Б.Г. Миркину за советы и конструктивные замечания. ■

Литература

1. Djukova E.V., Zhuravlev J.I., Rudakov K.V. Algebraic-logic synthesis of correct recognition procedures based on elementary algorithms // Computational Mathematics and Mathematical Physics, 1996, 36(8). – P. 1161–1167.
2. Zhuravlev J.I. Algebraic methods in recognition and classification problems // Pattern Recognition and Image Analysis, 1991, 1(1).
3. Кудинов П.Ю., Задача распознавания статистических таблиц // Доклады 14-й Всероссийской конференции «Математические методы распознавания образов» ММРО-2009. – М.: МАКС Пресс, 2009. – С. 552-555.
4. Кудинов П.Ю., Полежаев В.А. Динамическое обучение распознаванию статистических таблиц // Доклады 8-й Международной конференции «Интеллектуализация обработки информации» ИОИ-2010 (Республика Кипр, г. Пафос, 17-24 октября 2010). – М.: МАКС Пресс, 2010. – С. 512-515.
5. Utgoff P.E. An improved algorithm for incremental induction of decision trees // Machine Learning: Proceedings of the Eleventh International Conference, 1994. – P. 318-325.
6. Utgoff P.E., Berkman N.C., Clouse J.A. Decision tree induction based on efficient tree restructuring // Machine Learning, 1997, 29 (1). – P. 5-44.
7. Furnkranz J., Flach P.A. ROC n' rule learning – Towards a better understanding of covering algorithms // Machine Learning, 2005, 58 (1). – P. 39-77.
8. Breiman L., Schapire E. Random forests // Machine Learning, 2001, 45 (1). – P. 5-32.
9. Asuncion A., Newman D.J. UCI Machine Learning Repository. 2007. URL: http://www.ics.uci.edu/-mlearn/MLRepository.html.