Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра математических методов прогнозирования

Журавлёв Вадим Игоревич

О полных решающих деревьях в задаче регрессии

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Научный руководитель:

д.ф.-м.н., доцент

Е. В. Дюкова

Москва, 2018

Содержание

1 Введение

Одной из основных задач машинного обучения является задача обучения по прецедентам. Рассматривается следующая постановка этой задачи.

Исследуется множество объектов M. Объекты из M описываются системой признаков $\{x_1, \ldots, x_n\}$. Каждый объект S из M представим вектором длины n, в котором j-я координата равна значению признака x_j для объекта S. Задано некоторое числовое множество «ответов» Y и дана выборка объектов $T = \{S_1, \ldots, S_m\}$ из M такая, что для каждого объекта $S_i \in T$ известен «ответ» $y_i, y_i \in Y$. Объекты из T называются прецедентами или обучающими объектами. Требуется по выборке T построить алгоритм $A_T : M \longrightarrow Y$, восстанавливающий для каждого объекта S из M значение y из Y.

Актуальность рассматриваемой задачи заключается в том, что она возникает в целом ряде прикладных областей, таких как биология, геология, медицина, экономика, техника, банковская деятельность и т.д. Выделяют два основных типа задач обучения по прецедентам:

1. Задача классификации (classification). В этом случае «ответ» y для объекта S из M называется меткой класса. Возможны следующие варианты:
   - $Y = \{1, ..., N\}$ — классификация с N классами;
   - $Y = \{0, 1\}^N$ — классификация с N пересекающимися классами.
2. Задача восстановления регрессии (regression). В данном случае $Y = \mathbb{R}$ и «ответ» y для объекта S из M называется значением целевой переменной.

Одним из известных инструментов для решения задач классификации и восстановления регрессии являются деревья решений.

Процедура построения классического решающего дерева (РД) представляет собой итерационный процесс. Как правило, для построения очередной вершины дерева выбирается признак, наилучшим образом удовлетворяющий некоторому критерию ветвления. По значениям этого признака и осуществляется ветвление, далее указанная процедура повторяется для каждой из ветвей. Однако если при построении дерева несколько признаков удовлетворяют критерию ветвления в равной или почти равной мере, то выбор одного из них происходит случайным образом. При этом в зависимости от выбранного признака построенные деревья могут существенно отличаться как по составу используемых признаков, так и по своим распознающим качествам. Указанного недостатка лишена модель полного решающего дерева (ПРД) [1–5,12,14]. В ПРД на каждой

итерации строится так называемая полная вершина, которой соответствует набор признаков $\{x_{j_1}, \ldots, x_{j_q}\}, q \leq n$, в котором каждый признак удовлетворяет критерию ветвления в равной или почти равной мере. Затем, для каждого признака $x_{j_i}, i \in \{1, \ldots, q\}$, строится внутренняя (простая) вершина, из которой осуществляется ветвление. Модель ПРД первоначально исследовалась на задачах классификации по прецедентам и показала повышение качества решения по сравнению с наиболее известными методами синтеза регрессионных деревьев [1, 12].

В настоящей работе рассматривается задача восстановления регрессии.

Одним из первых алгоритмов, использующих регрессионное решающее дерево (РРД), является алгоритм CART (classification and regression trees) [7, 10, 16]. Этот алгоритм строит бинарное РРД с критерием ветвления, основанным на вычислении статистик. Алгоритм M5P [21], так же как и алгоритм CART, выполняет построение бинарных РРД. Более сложную конструкцию имеют алгоритмы классификации и восстановления регрессии Random Forest [8], REPTree [13] и Decision Stump [6]. Перечисленные алгоритмы строят бинарные деревья, за исключением алгоритма Decision Stump, который базируется на построении k-арных РРД, что является более сложной задачей.

Основной целью данной работы является построение и исследование полных РРД (ПРРД). В работе построены и протестированы алгоритмы построения ПРРД, а именно NBFRTree (non binary full regression tree), DFRTree (defined full regression tree) и RFRTree (random full regression tree). Алгоритм NBFRTree предназначен для задач с целочисленной информацией. При построении очередной простой вершины в алгоритме NBFRTree ветвление производится по всем значениям выбранного признака. Алгоритмы DFRTree и RFRTree предназначены для задач с вещественнозначной информацией. При построении очередной простой вершины в алгоритмах DFRTree и RFRTree из множества различных значений признака выбирается $k - 1$ пороговых значений, которые образуют k интервалов. Алгоритмы DFRTree и RFRTree различаются способом выбора пороговых значений. Построенные алгоритмы строят полные k-арные регрессионные деревья, где k – максимальное число ребер, выходящих из простых вершин дерева. В предоженных алгоритмах используется критерий ветвления, являющийся модификацией критерия ветвления алгоритма CART на случай k-арного дерева [7].

Проведено тестирование алгоритмов NBFRTree, RFRTree и DFRTree на реальных задачах. Из результатов тестирования следует, что на большинстве рассмотренных задач алгоритм RFRTree работает лучше алгоритмов NBFRTree и DFRTree, а также лучше других алгоритмов восстановления регрессии, участвовавших в тестировании, среди

которых алгоритмы Random Forest, REPTree, M5P, CART, Decision Stump.

Алгоритм RFRTree исследовался на задачах с пропущенными данными. Наилучшие результаты были получены при замене пропуска средним значением признака по всем обучающим объектам и при замене пропуска медианой тех значений признака, которые встречаются у ближайших соседей.

Работа состоит из шести разделов. Первый раздел — введение. Во втором разделе приведены основные понятия и дан обзор результатов, полученных ранее в данной области. Также приведено описание критерия ветвления алгоритма CART. В третьем разделе дано описание алгоритма NBFRTree, который выполняет построение ПРРД для задач с целочисленной информации. В четвертом разделе описаны алгоритмы построения ПРРД для задач с вещественнозначной информацией RFRTree и DFRTree. В пятом разделе представлены результаты тестирования алгоритмов NBFRTree, RFRTree и DFRTree на реальных задачах. Шестой раздел — заключение, в котором представлены полученные результаты.

2 Основные понятия, используемые при построении регресионных решающих деревьев. Обзор результатов, полученных в данной области

Рассмотрим основные понятия, используемые при построении k-арных РРД для задач с вещественнозначной информацией.

Обозначим через $\breve{T}$ и $X(\breve{T}) \subseteq \{x_1, ..., x_n\}$ рассматриваемые на текущей итерации (шаге) построения РРД подмножество обучающих объектов и подмножество признаков соответственно.

На первом шаге $\breve{T} = T, X(\breve{T}) = \{x_1, ..., x_n\}$. На текущем шаге построения дерева для каждого признака x из $X(\breve{T})$ выбирается множество из $k-1$ пороговых значений $\{z_1, \ldots, z_{k-1}\}$. Далее проводится разбиение $\breve{T}$ на k подвыборок и вычисляется оценка качества этого разбиения. В i-ую подвыборку попадают объекты из $\breve{T}$, для которых зачение признака x попадает в полуинтервал $(z_i, z_{i+1}], i \in \{0, \ldots, k-1\}$, где $z_0 = -\infty, z_k = +\infty$.

`Определение 1.` Оптимальным разбиением для признака называется такое разбиение, при котором достигается наилучшая оценка выбранного функционала качества.

`Определение 2.` Признак удовлетворяет критерию ветвления, если оптимальное разбиение для этого признака имеет максимальную оценку качества разбиения среди оптимальных разбиений для всех других признаков.

Среди всех признаков, удовлетворяющих критерию ветвления, выбирается только один признак.

Различные алгоритмы РРД в основном отличаются критерием ветвления, а также правилом останова ветвления.

Пусть признак x принимает l упорядоченных значений $\{r_1, r_2, \ldots, r_l\}$ и пусть для признака x было выбрано множество из $k-1$ различных друг от друга пороговых значений $\{r_{j_1}, \ldots, r_{j_{k-1}}\}$. На рис. 1 приведен пример ветвления из вершины x в РРД. В этом дереве $x \in X(\breve{T})$.

Алгоритм CART строит бинарное РРД (БРРД) и при выборе оптимального разбиения использует статистический подход к оценке качества разбиения (критерий ветвления, основанный на вычислении статистик). Опишем этот критерий.

Пусть $\breve{T} = \{S_{i_1}, \ldots, S_{i_u}\}, \breve{T}_R(x, a) = \{S_1^R, ..., S_q^R\}, \breve{T}_L(x, a) = \{S_1^L, ..., S_p^L\}$. При данном разбиении в правое и левое поддеревья попадает q и p объектов соответственно.

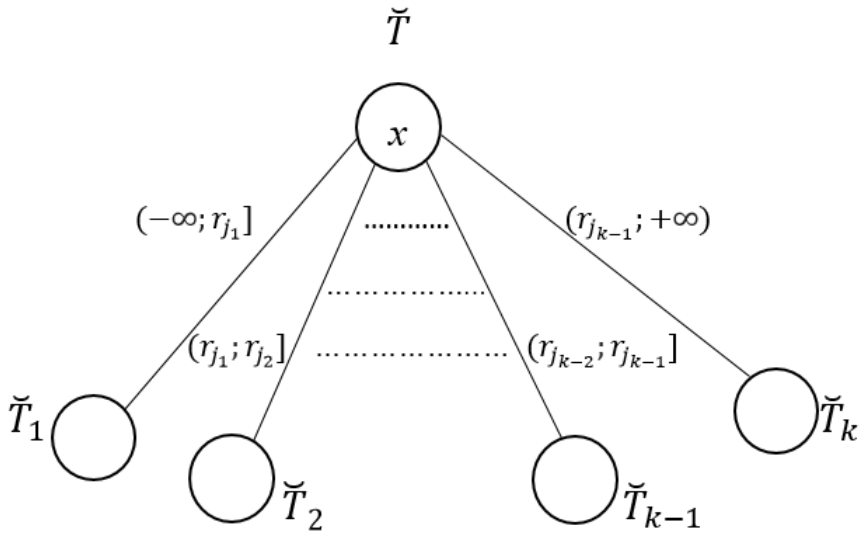

Рис. 1: **Пример ветвления из вершины x в k-арном РРД**

Пусть далее y_i^L и y_j^R — значения целевых переменных для объектов $S_i^L, i = 1, \ldots, p$, и $S_j^R, j = 1, ..., q$, соответственно. Введем обозначения:

$$\bar{y}_{\breve{T}} = \frac{1}{u} \sum_{t=1}^{u} y_{i_t} ;$$

$$\text{Variance} = \frac{1}{u} \sum_{t=1}^{u} (y_{i_t}^2) - \left[\frac{1}{u} \sum_{t=1}^{u} (y_{i_t}) \right]^2 ;$$

$$\text{SE}\,(x) = \frac{1}{u} \Big\{ \sum_{i=1}^{p} (y_i^L - \bar{y}_{\breve{T}_L})^2 + \sum_{j=1}^{q} (y_j^R - \bar{y}_{\breve{T}_R})^2 \Big\} ;$$

$$C(x) = \text{Variance} - \text{SE}(x).$$

Оптимальным считается разбиение с максимальным значением величины $C(x)$.

Построение очередной ветви в алгоритме CART прекращается, если величина $C(x)$ не превосходит наперед заданного числа δ.

Похожую на CART конструкцию имеет алгоритм M5P [16, 21]. Алгоритм M5P, так же как и алгоритм CART, выполняет построение БРРД, но использует энтропийный критерий ветвления.

Более сложную конструкцию имеют алгоритмы классификации и восстановления регрессии Random Forest [7, 8], REPTree [13, 20] и Decision Stump [6]. Алгоритм Decision Stump базируется на построении k-арных РРД, остальные алгоритмы строят БРРД. Среди перечисленных алгоритмов наиболее используемым является алгоритм Random Forest.

Random Forest — алгоритм машинного обучения, заключающийся в использовании комитета (ансамбля) БРРД (предложен Л. Брейманом и А. Катлер в 2001 г.). Для задачи восстановления регрессии в алгоритме Random Forest используется критерий ветвления, основанный на вычислениии статистик и процедура «бэггинг» . Основными параметрами в алгоритме Random Forest являются Q — число деревьев, $samplesize$ — размер подвыборки для «бэггинга» , $max\_features, 1 \leq max\_features < n$ — число признаков из которых выбирается признак для ветвления. Рекомендации по выбору этих параметров можно найти в [18]. Алгоритм построения случайного леса может быть представлен в следующем виде:

- Для $i = 1, 2, ..., Q$ (здесь Q — количество деревьев в ансамбле) выполнить
  - Выбрать подвыборку T_i обучающей выборки T размера $samplesize$ (с возвращением) — по ней строится дерево. Для каждого дерева — своя подвыборка.
  - Построить для T_i дерево, используя алгоритм построения РРД. На очередном шаге построения вершины для разбиения в дереве выбирается пара (x, a) наилучшим образом удовлетворяющая критерию ветвления (где x — признак из $max\_features$ случайных признаков, a — порог). Порог a выбирается как среднее арифметическое двух соседних упорядоченных значений признака x из обучающей подвыборки T_i. Вершина разбивается согласно паре (x, a) на две подвыборки — подвыборку T_{i_R} для объектов которой выполняется неравенство $x \geq a$, и подвыборку T_{i_L} для объектов которой выполняется неравенство $x < a$.
  - Дерево строится до полного исчерпания подвыборки и не подвергается процедуре отсечения ветвей.

Результат распознавания определяется путем усреднения значений целевой переменной по всем построенным БРРД. Таким образом, деревья компенсируют ошибки друг друга.

3 Алгоритм построения полного регрессионного решающего дерева NBFRTree (случай целочисленной информации)

Опишем алгоритм построения классического k-арного РРД NBRTree (Non binary regression tree) для лучшего понимания организации ветвления из простой вершины алгоритма NBFRTree. Алгоритм NBRTree строит классическое регрессионное решающее дерево. Главная особенность алгоритма NBRTree – это его k-арная структура. Ветвление по выбранному признаку x разбивает обучающие объекты на k подвыборок, где k — число различных значений признака.

Рассмотрим более подробно схему ветвления из вершины $x, x \in X(\breve{T})$ в алгоритме NBRTree.

Не ограничивая общности, будем считать, что признак x имеет значения из $\{0, 1, ..., k-1\}, k \geq 2$. В этом случае при построении дерева решений из вершины x выходят k дуг, помеченные числами из $\{0, 1, ..., k-1\}$. Пусть σ — метка одной из дуг, выходящих из вершины $x, \sigma \in \{0, 1, ..., k-1\}$. Для формирования нового текущего подмножества объектов и нового текущего множества признаков удаляются те объекты из $\breve{T}$ для которых значение признака x не равно σ, а также из множества признаков удаляется сам признак x. Для улучшения качества распознавания при построении ветви используется правило останова, которое описано в конце данного параграфа.

Положим

$$x^{\sigma} = \begin{cases} 1, & \text{если } x = \sigma, \\ 0, & \text{если } x \neq \sigma. \end{cases}$$

Пусть v – висячая вершина, порожденная ветвью дерева с внутренними вершинами $x_{j_1}, ..., x_{j_r}$ и пусть дуга, выходящая из вершины $x_{j_i}, i \in \{1, ..., r\}$, имеет метку σ. Пусть далее $\breve{T}(v)$ – текущее множество объектов, которые попали в вершину v. Вершине v ставится в соответствие пара $(B, w(v))$, где $w(v)$ равно среднему значению целевой переменной по всем объектам из $\breve{T}(v)$, а B – элементарная конъюнкция вида $x_{j_1}^{\sigma_1} ... x_{j_r}^{\sigma_r}$. Интервал истинности элементарной конъюнкции B обозначим через N_B.

Пусть S – распознаваемый объект. Для каждой висячей вершины $(B, w(v))$ выполняется проверка принадлежности описания распознаваемого объекта S интервалу истинности N_B. Если описание S принадлежит N_B, то объекту S ставим в соответствие значение целевой переменной $w(v)$. По построению описание может попасть только в

одну из висячих вершин.

На рис. 2 показано ветвление из вершины x в алгоритме NBRTree для $\breve{T} = \breve{T}_1 \cup \breve{T}_2 \cup ... \cup \breve{T}_k$, где k – множество различных значений признака x.

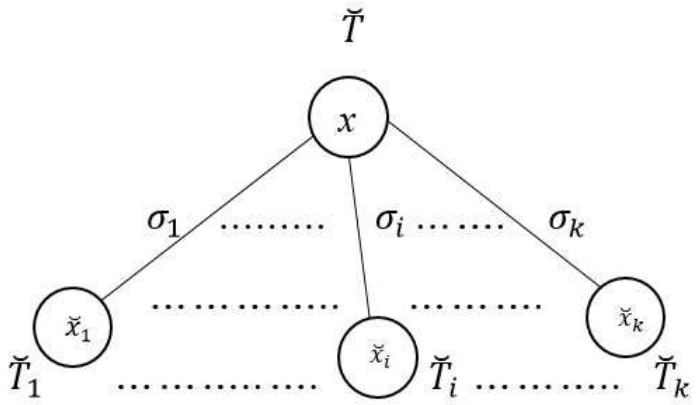

Рис. 2: **Пример ветвления из вершины x в алгоритме NBRTree**

В алгоритме NBFRTree используется идея ПРД, то есть при возникновении ситуации, когда два или более признака удовлетворяют критерию ветвления в равной или почти равной мере, в алгоритме NBFRTree проводится ветвление по каждому из этих признаков независимо.

Процедура распознавания объекта выполняется следующим образом. Пусть $V = \{v_1, ..., v_l\}$ – множество висячих вершин построенного дерева с соответствующими парами $(B_i, w(v_i)), i = 1, 2, ..., l, l \geq 1$. Для каждой висячей вершины v_i осуществляется проверка принадлежности описания объекта S интервалу истинности N_{B_i}.

Положим

$$I_{B_i} = \begin{cases} 1, & \text{если описание объекта } S \in N_{B_i}, \\ 0, & \text{в противном случае.} \end{cases}$$

Объекту S ставится в соответствие значение целевой переменной

$$W = \frac{\sum\limits_{i=1}^{l} w(v_i) I_{B_i}}{\sum\limits_{i=1}^{l} I_{B_i}}$$

На рис. 3 показано ветвление из полной вершины $\{x_{j_1}, ..., x_{j_r}\}$ в алгоритме NBFRTree. Ветвление из простых вершин $x_{j_1}, ..., x_{j_r}$ в алгоритме NBFRTree производится также, как в алгоритме NBRTree.

Опишем критерий ветвления, используемый в алгоритмах NBRTree и NBFRTree.

Пусть $\breve{T}_i = S_1^i, ..., S_{u_i}^i, y_t^i$ – значение целевой переменной обучающего объекта $S_t^i, t \in \{1, 2, ..., u_i\}$ и пусть рассматриваемый признак x принимает k значений. Обучающая выборка $\breve{T}_i$ разбивается по этому признаку на k подвыборок $\breve{T}_{i_1}, ..., \breve{T}_{i_k}$. Вычис-

ляются величины

$$SE(x,k) = \frac{1}{u_i}\Big\{ \sum_{S_t^i \in \breve{T}_{i_1}} (y_t^i - \bar{y}_{\breve{T}_{i_1}})^2 + ... + \sum_{S_t^i \in \breve{T}_{i_k}} (y_t^i - \bar{y}_{\breve{T}_{i_k}})^2 \Big\} \tag{1}$$

$$C(k,x) = Variance - SE(x,k). \tag{2}$$

При $k = 2$ описанный критерий совпадает с критерием ветвления алгоритма CART (см. раздел 2).

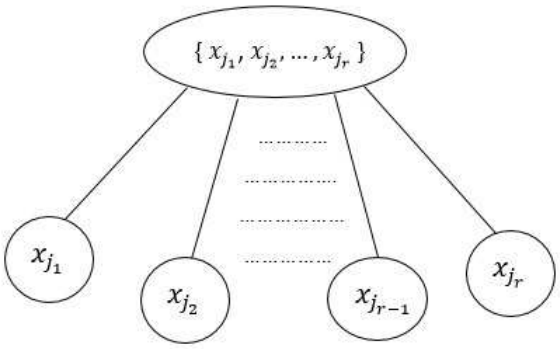

Рис. 3: **Пример ветвления из полной вершины $\{x_{j_1}, ..., x_{j_r}\}$ в алгоритме NBFRTree**

В алгоритме NBRTree для ветвления выбирается один признак $\hat{x}$, такой что $\hat{x} = \arg\max_{x \in X(\breve{T})} C(k,x)$.

В алгоритме NBFRTree признаки для ветвления выбираются иначе. Пусть $C_{min} = \min_{x \in X(\breve{T})} C(k,x)$ и $C_{max} = \max_{x \in X(\breve{T})} C(k,x)$. Сначала для каждого признака $x \in X(\breve{T}_i)$ вычисляется величина $C(k,x) = Variance - SE(x,k)$. Далее значение $C(k,x)$ нормируется и вычисляется

$$C^*(k,x) = \frac{C(k,x) - C_{min}}{C_{max} - C_{min}}.$$

Для построения полной вершины выбираются те признаки из $X(\breve{T}_i)$, для которых $0.75 \leq C^*(k,x) \leq 1$. В случае, когда $C_{max} = C_{min}$, разбиение производится по всем признакам из $X(\breve{T}_i)$.

Построение ветви прекращается, если разность между минимальным и максимальным целевыми переменными в данной вершине не превосходит наперед заданного ε (параметр останова).

4 Алгоритмы построения полных регрессионных решающих деревьев DFRTree и RFRTree (случай вещественнозначной информации)

В алгоритмах DFRTree и RFRTree при построении регрессионного дерева на каждой итерации строится так называемая полная вершина, которой соответствует набор пар $Z = \{(x_{i_1}, D(x_{i_1})), \ldots, (x_{i_t}, D(x_{i_t}))\}$, где $D(x_{i_j}), i_j \in \{1, ..., n\}, j \in \{1, ..., t\}$ — множество порогов для признака x_{i_j} . В этом наборе каждая пара $(x_{i_j}, D(x_{i_j}))$ удовлетворяет критерию ветвления в равной или почти равной мере. Затем для каждой такой пары строится «простая» внутренняя вершина x_{i_j}, из которой осуществляется ветвление с числом дуг, определяемым мощностью $D(x_{i_j})$.

При построении очередной простой вершины в алгоритмах DFRTree и RFRTree из множества различных значений признака $x, x \in X(\breve{T})$ (за исключением наибольшего значения) выбирается $k-1$ пороговых значений. Алгоритмы DFRTree и RFRTree различаются способом выбора пороговых значений для признака x.

На рис. 4 показано ветвление из полной вершины в алгоритмах DFRTree и RFRTree.

Пусть далее g – число различных значений признака x. В алгоритме DFRTree из множества значений признака x выбирается $k-1, 2 \leq k \leq g$, порогов $z_1, z_2, \ldots, z_{k-1}$ таким образом, чтобы при разбиении в каждый из k полуинтервалов $(-\infty, z_1]$, $(z_1, z_2], \ldots,$ $(z_{k-1}, +\infty)$ попало примерно одинаковое количество объектов из $X(\breve{T})$.

В алгоритме RFRTree из множества значений признака $x, x \in X(\breve{T})$ (за исключением наибольшего значения) случайным образом выбирается и сортируется $k-1$ различных порогов $z_1, \ldots, z_{k-1}$. Наибольшее значение признака x отсекается для того, чтобы при разбиении не возникало ситуации, когда в вершину не попал ни один объект. Далее вычисляется оценка качества данного разбиения. Процедура выбора пороговых значений выполняется $h, h \in \mathbb{N}$ раз. Для признака x выбирается разбиение, удовлетворяющее критерию ветвления наилучшим образом. Если положить $k = g$, то алгоритм RFRTree совпадает с алгоритмом NBFRTree, описанным в разделе 3.

Рассмотрим более подробно схему ветвления из вершины $x, x \in X(\breve{T})$ в алгоритмах DFRTree и RFRTree для случая вещественнозначной информации.

Пусть $\{z_1, z_2, \ldots, z_{k-1}\}, k \geq 2$, – множество отсортированных пороговых значений для признака x, встречающихся в описании объектов из $\breve{T}$. В этом случае при построении дерева решений из вершины x выходит k дуг, помеченных полуинтервалами из

$\{(-\infty, z_1], (z_1, z_2], \ldots, (z_{k-1}, +\infty)\}$.

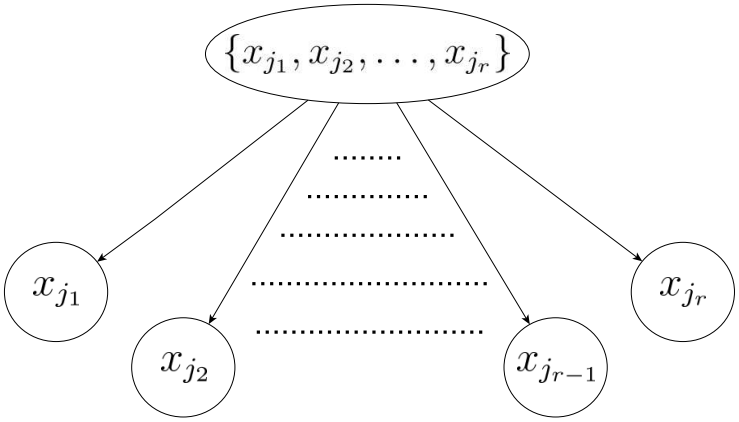

Рис. 4: **Пример ветвления из полной вершины в алгоритмах DFRTree и RFRTree**

Пусть σ – метка одной из дуг, выходящих из вершины $x, \sigma \in \{(-\infty, z_1], (z_1, z_2], \ldots, (z_{k-1}, +\infty)\}$. Для формирования нового текущего подмножества объектов и нового текущего множества признаков удаляются те объекты из $\breve{T}$, для которых значение признака x не попадает в полуинтервал σ, а также из множества признаков удаляется сам признак x. Для улучшения качества распознавания при построении ветви используется правило останова, которое описано в конце данного параграфа.

Положим

$$x^{\sigma} = \begin{cases} 1, & \text{если } x \in \sigma, \\ 0, & \text{если } x \notin \sigma. \end{cases}$$

Пусть v – вершина, порожденная ветвью дерева с внутренними вершинами $x_{j_1}, \ldots, x_{j_r}$, и пусть дуга, выходящая из вершины x_{j_i}, имеет метку $\sigma_i, i \in \{1, \ldots, r\}$. Пусть далее $\breve{T}(v)$ — текущее множество объектов, которые попали в вершину v. Вершине v ставится в соответствие пара $(B, w(v))$, где $w(v)$ равно среднему значению целевой переменной по всем объектам из $\breve{T}(v)$, а B – элементарная конъюнкция вида $x_{j_1}^{\sigma_1} \ldots x_{j_r}^{\sigma_r}$. Интервал истинности элементарной конъюнкции B обозначим через N_B.

Пусть S — распознаваемый объект. Для каждой висячей вершины $(B, w(v))$ выполняется проверка принадлежности описания распознаваемого объекта S интервалу истинности N_B. Если описание S принадлежит N_B, то объекту S ставим в соответствие значение целевой переменной $w(v)$.

На рис. 5 приведён пример ветвления из простой вершины x в алгоритмах DFRTree и RFRTree.

В алгоритмах DFRTree и RFRTree используется идея ПРРД, т. е. при возникновении

ситуации, когда два или более признака удовлетворяют критерию ветвления в равной или почти равной мере, то ветвление проводится по каждому из этих признаков независимо.

Процедура распознавания объекта S выполняется следующим образом. Пусть $V = \{v_1, \ldots, v_l\}$ — множество висячих вершин построенного дерева с соответствующими парами $(B_i, w(v_i)), i = 1, 2, ..., l, l \geq 1$. Для каждой висячей вершины v_i осуществляется проверка принадлежности описания объекта S интервалу истинности N_{B_i}.

Положим

$$I_{B_i} = \begin{cases} 1, & \text{если описание объекта } S \in N_{B_i}\,; \\ 0, & \text{в противном случае.} \end{cases}$$

Объекту S ставится в соответствие значение целевой переменной

$$W = \frac{\sum_{i=1}^{l} w(v_i) I_{B_i}}{\sum_{i=1}^{l} I_{B_i}}\,.$$

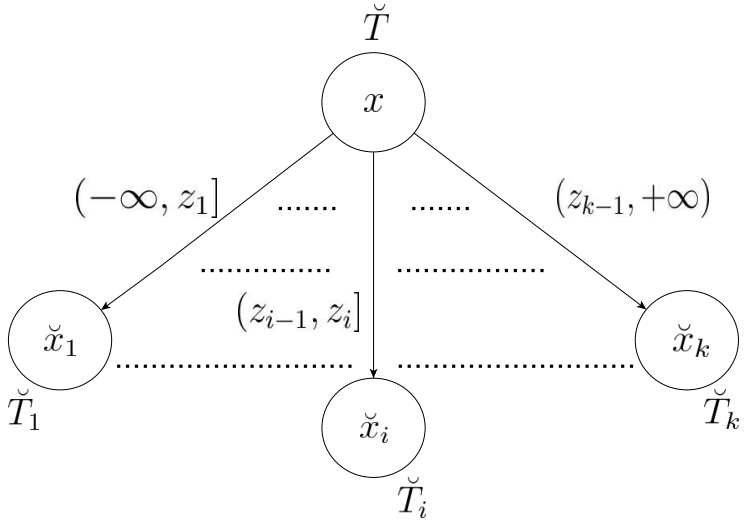

Рис. 5: **Пример ветвления из простой вершины x в алгоритмах DFRTree и RFRTree**

Опишем критерий ветвления, используемый в алгоритмах DFRTree и RFRTree.

Пусть $\breve{T}_i = S_1^i, ..., S_{u_i}^i, y_t^i, i \in \{0, 1, \ldots, m\}$ – значение целевой переменной обучающего объекта $S_t^i, t \in \{1, 2, \ldots, u_i\}$ и пусть из множества отсортированных значений признака x (за исключением наибольшего значения) было выбрано $k-1$ различных пороговых значений $z_1, z_2, \ldots, z_{k-1}$. Таким образом, значения признака x разбиваются на k полуинтервалов $\{(-\infty, z_1], (z_1, z_2], \ldots, (z_{k-1}, +\infty)\}$. Обучающая выборка $\breve{T}_i$ разбивается по этому признаку на k подвыборок $\breve{T}_{i_1}, ..., \breve{T}_{i_k}$. Далее вычисляются величины $SE(k, x)$ и $C(k, x)$ по формулам 1–2.

Выбор признаков для построения полной вершины осуществляется также, как в алоритме NBFRTree, описанном в предыдущем разделе. Значение $C(k,x)$ нормируется и вычисляется

$$C^*(k,x) = \frac{C(k,x) - C_{min}}{C_{max} - C_{min}}.$$

Для построения полной вершины выбираются те признаки из $X(\breve{T}_i)$, для которых $0.7 \leq C^*(k,x) \leq 1$. В случае, когда $C_{max} = C_{min}$, разбиение производится по всем признакам из $X(\breve{T}_i)$. Пусть признак x принимает e различных значений. Если в описанном критерии ветвления положить $k = e - 1$, то алгоритм RFRTree совпадёт с алгоритмом NBFRTree.

При $k = 2$ описанный критерий также совпадает с критерием ветвления алгоритма CART (см. раздел 2).

Построение ветви прекращается, если разность между минимальным и максимальным целевыми переменными в данной вершине не превосходит наперед заданного ε (параметр останова).

С проблемой обработки пропусков в данных приходится сталкиваться при проведении разнообразных социологических, экономических и статистических исследований. Традиционными причинами, приводящими к появлению пропусков, являются невозможность получения или обработки, искажение или сокрытие информации. В результате на вход программ анализа собранных данных поступают неполные сведения. Существует два варианта действий: исключить объекты, содержащие пропуски, или заполнить тем или иным образом пропуски и анализировать полученную заполненную матрицу.

Метод исключения объектов с пропусками в описании (некомплектных) состоит в том, что все объекты, содержащие пропуски, исключают из рассмотрения и в дальнейшем анализируют новую, редуцированную матрицу данных. Когда выборка содержит достаточное число комплектных объектов, такой подход следует признать наиболее целесообразным.

Однако на практике распространена ситуация, когда наличие даже небольшого числа случайно (или, как минимум, без явной закономерности) распределенных пропусков — при формально большой размерности данных — приводит к резкому уменьшению числа комплектных наблюдений. Потому необходимо на этапе первичной обработки данных заполнить пропуски в уже имеющихся данных, для того чтобы восстановить

исходную зависимость.

Заполнение пропусков позволяет не только получить дополнительную информацию (предсказанные значения), но и сохранить уже имеющуюся, часто очень важную и полученную ценой значительных усилий информацию, за счет сохранения наблюдений изначально содержавших пропуски. Помимо очевидных достоинств, заполнение пропусков, как способ решения проблемы недостающей информации имеет несколько недостатков, которые нельзя не учитывать:

1. Использование для предсказания пропусков имеющихся полных данных искажает структуру результирующих данных (после заполнения пропусков в данных), которая смещается в сторону структуры только полных наблюдений;

2. Искусственная подстановка пропусков вносит в массив определенную долю искусственных данных, которые в свою очередь приводят к смещению значимости получаемых на их основе результатов.

Для заполнения пропусков в обучающих данных использовались такие числовые характеристики множества значений признака как среднее арифметическое, медиана и мода.

Идея заключается в замене отсутствующих данных средним значением, медианой и модой по всем объектам из обучающей выборки. Это один из наиболее распространенных методов. Но у этого метода есть недостатки. Использование константных значений для замены отсутствующих данных изменяет характеристику исходного набора данных. Это может повлечь за собой потери зависимостей между признаками, что приведёт к смещению ответов алгоритма.

Также для восстановления пропущенных данных был исследован метод k ближайших соседей. Этот метод использует алгоритм k-ближайших соседей для замены пропущенных данных. Данный метод основывается на следующем предположении: близкие объекты по известным значениям признаков также близки в признаках, значение которых может быть пропущено. Основные преимущества этого метода:

- он может оценивать как качественные признаки (наиболее частое значение среди k ближайших соседей), так и количественные признаки (среднее значение среди k ближайших соседей)

- нет необходимости создавать прогностическую модель для каждого признака с пропущенными данными.

В этом пропущенное значение заменяется средним значением, медианой или модой по «ближайшим» k объектам. В данной работе параметр k был положен равным 10.

При использовании данных без пропусков необходимо создать пропущенные значения искусственно. В работах [11, 15] некоторые значения в признаковом описании объектов удаляют случайным образом.

Для более подробного анализа методов было произведено исследование с разным количеством пропущенных данных. Пропуски генерировались в данных случайным образом в следующем соотношении: в 10%, 20%, 30%, 40%, 50% от числа обектов в выборке. В признаковом описании отдельно взятого объекта пропущенное значение генерировалось с вероятностью 0.05%.

5 Тестирование алгоритмов построения полных регрессионных решающих деревьев NBFRTree, DFRTree и RFRTree

Алгоритмы NBFRTree, DFRTree и RFRTree были протестированы на 15 реальных задачах из ресурса UCI [19]. Список задач, на которых производилось тестирование алгоритмов: Data1 — Fertility; Data2 — Autos; Data3 — Servo; Data4 — Breast Cancer Wisconsin wpbc; Data5 — Yacht Hydrodynamics; Data6 — Auto MPG; Data7 — Housing; Data8 — Forest Fires; Data9 — Breast Cancer Wisconsin wdbc; Data10 — Geographical Original of Music Data Set latitude; Data11 — Geographical Original of Music Data Set longitude; Data12 — Airfoil Self-Noise; Data13 — Red Wine Quality; Data14 — White Wine Quality; Data15 — Combined Cycle Power Plant.

Data	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
ε	0.01	1000	0.1	1	0.01	0.5	0.1	1	1	80	170	0	1	1	4

Таблица 1: Оптимальное значение ε

Значение параметра останова ε для каждой задачи определялось эмпирически. Для разных значений ϵ производилась перекрёстная проверка. В результате выбиралось то значение ε, при котором достигался наилучший результат алгоритма. В табл. 1 приведено оптимальное значение ε для каждой из рассмотренных задач.

Для оценки качества работы алгоритмов была применена кросс-валидация по k фолдам. Исходные данные разбивались на k подвыборок,$k \geq 2$. Затем на $k-1$ подвыборке производилось обучение алгоритма, а оставшаяся подвыборка использовалась для тестирования. Процедура повторялась k раз. В итоге каждая из k подвыборок использовалась для тестирования.

Для оценки эффективности алгоритмов использовались величины MAE (Mean Absolute Error – средняя абсолютная ошибка) и RMSE (Root Mean Squared Error – корень среднеквадратичной ошибки), вычисляемые соответственно следующим образом:

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|y_i - h_i|, \qquad RMSE = \frac{1}{\sqrt{m}}\sqrt{\sum_{i=1}^{m}(y_i - h_i)^2},$$

где y_i – значения целевых переменных, а h_i – значения, выданные алгоритмом.

Задача	% пропусков	moda	median	mean	knn_moda	knn_median	knn_mean
Data1		0.178	0.179	0.176	**0.147**	0.154	0.181
Data2		1216.604	1339.452	1181.405	1117.886	**1072.071**	1100.271
Data4		0.335	0.313	0.316	0.336	**0.312**	0.343
Data6	**10%**	2.035	2.013	**1.961**	2.060	2.081	1.998
Data7		2.495	2.447	2.574	2.418	**2.352**	2.525
Data8		21.569	21.551	**16.168**	20.077	20.265	20.831
Data12		2.260	2.254	1.933	2.235	2.278	**1.920**
Data1		0.183	0.219	0.186	0.194	**0.152**	0.209
Data2		1485.755	1466.818	1201.499	1234.504	1218.452	**1087.277**
Data4		0.319	0.326	0.318	0.336	0.346	**0.313**
Data6	**20%**	2.111	2.207	**2.013**	2.087	2.057	2.017
Data7		2.670	2.676	2.724	2.666	2.689	**2.613**
Data8		20.235	20.232	**16.979**	18.044	17.482	18.261
Data12		2.536	2.466	2.039	2.666	2.460	**1.991**
Data1		0.194	0.185	**0.144**	0.217	0.178	0.160
Data2		1329.397	1384.381	1509.848	1181.789	**1158.555**	1260.753
Data4		0.338	0.334	0.336	0.347	**0.330**	0.346
Data6	**30%**	2.094	2.106	2.212	2.082	**2.024**	2.150
Data7		**2.551**	2.603	2.673	2.667	2.732	2.701
Data8		19.405	19.946	17.920	18.170	16.859	**16.535**
Data12		2.759	2.797	2.053	2.843	2.741	**2.030**
Data1		0.239	0.210	0.215	0.199	0.186	**0.172**
Data2		1484.635	1463.036	1406.215	1093.569	**1074.748**	1149.354
Data4		0.338	**0.328**	0.334	0.337	0.336	0.332
Data6	**40%**	2.191	2.250	2.196	**2.116**	2.229	2.192
Data7		2.826	**2.674**	2.794	2.828	2.733	2.721
Data8		18.935	22.676	**15.982**	18.929	20.470	17.812
Data12		2.835	2.694	**2.180**	2.676	2.874	2.213
Data1		0.184	0.193	0.195	0.179	**0.158**	0.182
Data2		1635.385	1540.635	1632.258	1234.851	1185.656	**1173.064**
Data4		0.332	0.341	**0.285**	0.322	0.323	0.313
Data6	**50%**	2.419	2.441	**2.071**	2.470	2.356	2.184
Data7		2.871	2.952	3.039	2.870	2.919	**2.840**
Data8		20.615	18.343	21.031	18.412	**15.831**	17.975
Data12		3.083	3.107	2.216	3.226	2.979	**2.188**

Таблица 2: Результаты тестирования алгоритма RFRTree с пропусками в данных (Качество работы оценивается функционалом качества MAE)

Задача	% пропусков	moda	median	mean	knn_moda	knn_median	knn_mean
Data1		0.348	0.336	0.354	**0.308**	0.321	0.359
Data2		1964.347	2210.143	1871.573	1754.078	**1701.705**	1767.363
Data4		0.400	**0.378**	0.397	0.398	0.396	0.410
Data6	**10%**	2.725	2.772	**2.711**	2.815	2.857	2.759
Data7		3.878	3.849	3.860	**3.616**	3.660	3.694
Data8		59.722	62.570	**48.085**	50.663	53.954	68.132
Data12		3.024	2.969	**2.684**	2.920	3.006	2.687
Data1		0.308	0.353	0.333	0.340	**0.289**	0.364
Data2		2378.509	2342.259	1784.022	1881.569	1833.236	**1651.458**
Data4		**0.389**	0.411	0.403	0.403	0.411	0.418
Data6	**20%**	2.977	3.025	**2.785**	2.917	2.819	2.808
Data7		4.099	4.217	4.115	4.279	4.062	**4.011**
Data8		50.750	56.894	48.276	**46.811**	48.741	51.720
Data12		3.328	3.269	2.751	3.510	3.249	**2.712**
Data1		0.335	0.310	**0.279**	0.370	0.323	**0.279**
Data2		2061.311	2121.023	2294.071	1874.348	**1793.498**	1987.870
Data4		0.423	0.421	**0.395**	0.419	0.404	0.418
Data6	**30%**	2.907	2.939	3.148	2.91	**2.745**	2.994
Data7		4.077	3.992	**3.953**	4.195	4.263	3.993
Data8		50.742	55.634	55.831	45.211	**42.062**	52.134
Data12		3.598	3.627	**2.786**	3.775	3.630	2.792
Data1		0.392	0.361	0.356	0.358	0.348	**0.274**
Data2		2254.190	2280.334	2125.209	1830.004	1762.762	**1738.641**
Data4		0.393	0.394	0.409	0.401	**0.394**	0.398
Data6	**40%**	3.062	3.055	2.967	**2.903**	3.055	2.957
Data7		4.227	**3.955**	4.229	4.345	4.274	4.137
Data8		55.716	63.384	**45.156**	50.267	56.901	53.500
Data12		3.759	3.540	**3.002**	3.556	3.780	3.106
Data1		0.326	0.312	0.357	0.325	**0.301**	0.350
Data2		2718.018	2566.649	2569.519	1953.362	1958.479	**1838.618**
Data4		0.389	0.400	0.380	0.389	**0.379**	0.409
Data6	**50%**	3.362	3.344	**2.939**	3.396	3.269	2.975
Data7		4.383	4.356	4.718	**4.199**	4.384	4.337
Data8		66.335	47.075	67.333	50.103	**42.624**	59.202
Data12		4.126	4.100	3.038	4.230	3.937	**2.928**

Таблица 3: Результаты тестирования алгоритма RFRTree с пропусками в данных (Качество работы оценивается функционалом качества RMSE)

Из табл. 2 и табл. 3 видно, что наилучшие результаты на большинстве задач с пропусками достигаются при использовании метода k ближайших соседей. Также хорошие результаты показал метод с заменой пропуска на среднее значение по признаку.

Для алгоритма RFRTree был дополнительно исследован параметр h, описанный в разделе 3, - число выполнений процедуры выбора пороговых значений. В табл. 4 и табл. 5 приведены результаты тестирования при $h = 1, 5, 10, 20, 50, C^* = 0.85$ и $k = 6$.

Задача	Размер $m \times n$	$h = 1$	$h = 5$	$h = 10$	$h = 20$	$h = 50$
Data1	100×10	0.210	0.182	**0.173**	0.184	0.201
Data2	160×25	1296.583	1280.347	**1240.264**	1253.730	1280.794
Data3	167×4	0.401	0.265	0.265	0.272	**0.253**
Data4	198×33	0.298	0.288	**0.268**	0.295	**0.268**
Data5	308×6	4.341	3.335	2.611	2.400	**2.164**
Data6	398×8	1.997	**1.989**	2.057	2.079	2.103
Data7	506×13	3.312	2.934	**2.700**	2.746	2.836
Data8	517×12	20.911	19.567	**18.014**	19.940	20.300
Data9	569×30	**0.036**	0.040	0.049	0.045	0.049
Data10	1059×68	13.615	13.562	**13.447**	13.759	13.568
Data11	1059×68	35.309	**35.271**	35.400	35.466	35.962
Data12	1503×5	2.662	2.534	**2.370**	2.396	2.412
Data13	1599×11	0.393	0.387	0.384	**0.371**	0.372
Data14	4898×11	0.388	0.385	**0.384**	0.390	0.396
Data15	9568×4	3.421	3.135	3.112	3.111	**3.075**
Отклонение от $h = 1$			+6.54%	+8.22%	+7.26%	+6.65%

Таблица 4: Результаты тестирования алгоритма RFRTree при различных значениях параметра h (качество работы оценивается функционалом качества MAE)

Как видно из табл. 4 и табл. 5, на большинстве задач наилучшие результаты алгоритм RFRTree достиг при использовании параметра $h = 10$, что на 8.22% лучше, чем при $h = 1$.

В табл. 6 представлены результаты тестирования алгоритмов RFRTree, DFRTree и NBFRTree. Алгоритмы RFRTree и DFRTree были протестированы с параметрами $h = 10, C^* = 0.85, k = 6$ и $C^* = 0.85, k = 6$ соответственно.

`Замечание.` Алгоритм NBFRTree предназначен для задач с целочисленной информацией (см. раздел 3), поэтому в задачах, в которых присутствовали признаки, при-

Задача	Размер $m \times n$	$h = 1$	$h = 5$	$h = 10$	$h = 20$	$h = 50$
Data1	100×10	0.372	**0.364**	0.338	0.373	0.409
Data2	160×25	1993.421	**1916.560**	1928.313	1957.921	1918.817
Data3	167×4	0.923	0.507	0.535	0.531	**0.481**
Data4	198×33	0.546	0.537	**0.517**	0.542	**0.517**
Data5	308×6	8.184	5.737	4.617	3.960	**3.395**
Data6	398×8	2.823	**2.775**	2.811	2.935	2.907
Data7	506×13	4.816	4.319	**3.885**	4.027	3.933
Data8	517×12	54.588	53.834	**48.460**	53.923	57.153
Data9	569×30	0.181	**0.172**	0.214	0.193	0.211
Data10	1059×68	17.021	**17.001**	16.944	17.120	17.022
Data11	1059×68	**44.534**	44.745	45.322	45.832	46.340
Data12	1503×5	3.435	3.291	**3.125**	3.143	3.184
Data13	1599×11	0.700	0.689	0.699	**0.674**	0.679
Data14	4898×11	0.712	0.708	**0.701**	0.708	0.718
Data15	9568×4	4.482	4.122	4.129	4.122	**4.101**
Отклонение от $h = 1$			+6.54%	+8.22%	+7.26%	+6.65%

Таблица 5: Результаты тестирования алгоритма RFRTree при различных значениях параметра h (качество работы оценивается функционалом качества RMSE)

нимающие вещественнозначные значения, была применена процедура перекодирования вещественнозначных значений признака в целочисленные. Производилась она следующим образом. Пусть $\{c_1, ..., c_u\}$ – множество различных значений признака x, $c_{i+1} > c_i, 1 \leq i \leq u - 1$. Выбирается t порогов, для признака x, делящих обучающую выборку по этому признаку на t равных частей. Для однообразия эксперимента t положили равным пяти.

Из табл. 6 видно, что алгоритм RFRTree лучше алгоритма DFRTree в среднем на 2.5% по функционалу MAE и на 0.36% по функционалу RMSE, а также лучше алгоритма NBFRTree в среднем на 7.56% по функционалу MAE и на 1.91% по функционалу RMSE.

Алгоритм RFRTree с наилучшими параметрами сравнивался с алгоритмами CART и Random Forest из библиотеки sklearn языка Python, а также с алгоритмами Decision Stump (DS), M5P и REPTree из свободного программного обеспечения для анализа данных WEKA.

Для всех задач применялась кросс-валидация по 10 фолдам. Для большей надежно-

сти эксперимента кросс-валидация по 10 фолдам производилась 10 раз, после каждой итерации выборка перемешивалась.

Задача	Размер $m \times n$	RFRTree		DFRTree		NBFRTree	
		MAE	RMSE	MAE	RMSE	MAE	RMSE
Data1	100 × 10	0.170	0.380	**0.130**	0.297	0.156	**0.294**
Data2	160 × 25	**1235.656**	1961.113	1330.200	**1956.234**	1491.946	2261.934
Data3	167 × 4	0.569	0.817	0.467	0.649	**0.264**	**0.414**
Data4	198 × 33	**0.305**	0.542	0.353	0.578	0.351	**0.508**
Data5	308 × 6	1.651	2.490	1.924	3.257	**0.808**	**1.474**
Data6	392 × 8	2.122	2.995	**2.037**	**2.775**	2.389	3.214
Data7	506 × 13	**2.898**	4.432	2.917	**4.426**	3.221	4.870
Data8	517 × 12	**18.071**	**54.027**	19.132	55.933	23.598	68.581
Data9	569 × 30	**0.042**	**0.185**	0.058	0.221	0.077	0.270
Data10	1059 × 68	13.762	17.332	**13.399**	**16.935**	13.626	17.543
Data11	1059 × 68	**36.744**	**47.213**	38.156	48.157	37.627	48.152
Data12	1503 × 5	2.429	3.181	**2.170**	**2.832**	2.613	3.347
Data13	1599 × 11	**0.393**	**0.713**	0.420	0.735	0.436	0.755
Data14	4898 × 11	**0.417**	**0.743**	0.423	0.757	0.462	0.811
Data15	9568 × 4	**3.305**	**4.332**	3.318	4.371	4.028	5.140
Отклонение от RFRTree				-2.50%	-0.36%	-7.56%	-1.91%

Таблица 6: Результаты тестирования алгоритмов RFRTree, DFRTree и NBFRTree

В табл. 7 и табл. 8 приведены результаты тестирования алгоритма RFRTree с другими известными алгоритмами. Из этих таблиц видно, что на 11 из 15 реальных задач с функционалом качества MAE наилучшие результаты показал алгоритм RFRTree, на трёх — Random forest и на одной — CART. На 10 из 15 реальных задач с функционалом качества RMSE наилучшие результаты показал алгоритм RFRTree, на трёх — Random Forest, на одной — алгоритмы CART и Desicion Stump. Таким образом, наилучшие результаты на большинстве задач показал алгоритм RFRTree. На задачах, на которых наилучшие результаты показал RFRTree, алгоритм RFRTree оказался лучше в среднем на 27% (функционал MAE) и на 16% (функционал RMSE) по сравнению с алгоритмом Random Forest.

Задача	Размер m × n	RFRTree	RF	CART	DS	REPTree	M5P
Data1	100 × 10	**0.100**	0.193	0.185	0.195	0.195	0.211
Data2	160 × 25	**1141.085**	1329.765	1472.863	2473.688	1667.338	1919.988
Data3	167 × 4	0.396	0.226	**0.209**	0.647	0.316	0.589
Data4	198 × 33	**0.269**	0.325	0.295	0.324	0.340	0.342
Data5	308 × 6	1.462	**0.536**	0.622	5.360	0.818	2.470
Data6	392 × 8	**1.717**	2.539	3.016	4.084	2.524	2.630
Data7	506 × 13	**2.240**	2.517	3.128	5.420	2.894	3.300
Data8	517 × 12	**15.417**	22.258	23.620	18.734	19.596	18.524
Data9	569 × 30	**0.034**	0.083	0.074	0.166	0.089	0.144
Data10	1059 × 68	**12.193**	12.810	15.169	13.927	13.754	13.748
Data11	1059 × 68	**33.114**	34.740	40.655	40.297	37.612	37.634
Data12	1503 × 5	1.671	**1.355**	1.884	5.030	2.332	3.099
Data13	1599 × 11	**0.292**	0.426	0.438	0.571	0.515	0.527
Data14	4898 × 11	**0.276**	0.447	0.463	0.673	0.572	0.566
Data15	9568 × 4	2.814	**2.436**	2.977	7.399	2.930	3.100
Отклонение от RFRTree			-24.02%	-32.77%	-126.61%	-41.83%	-72.01%

Таблица 7: Результаты тестирования алгоритма RFRTree с другими известными алгоритмами (качество работы оценивается функционалом качества MAE)

Задача	Размер m × n	RFRTree	RF	CART	DS	REPTree	M5P
Data1	100 × 10	**0.198**	0.311	0.386	0.314	0.346	0.328
Data2	160 × 25	**1796.030**	2024.363	2273.114	3787.171	2690.807	3162.889
Data3	167 × 4	0.618	0.453	**0.407**	1.019	0.754	0.975
Data4	198 × 33	0.513	0.414	0.535	**0.411**	0.444	0.416
Data5	308 × 6	2.210	**1.011**	1.206	7.536	1.528	4.650
Data6	392 × 8	**2.438**	3.368	4.084	5.256	3.576	3.633
Data7	506 × 13	**3.460**	3.678	4.977	7.460	4.323	4.835
Data8	517 × 12	**43.022**	57.676	73.348	63.962	64.590	63.840
Data9	569 × 30	**0.164**	0.196	0.264	0.313	0.230	0.237
Data10	1059 × 68	**15.480**	16.947	22.284	17.474	17.327	17.310
Data11	1059 × 68	**42.450**	44.550	57.633	50.202	49.163	47.973
Data12	1503 × 5	2.194	**1.884**	2.618	6.343	3.082	3.980
Data13	1599 × 11	**0.556**	0.605	0.770	0.735	0.671	0.672
Data14	4898 × 11	**0.552**	0.636	0.816	0.812	0.747	0.727
Data15	9568 × 4	3.731	**3.447**	4.401	9.035	3.997	4.078
Отклонение от RFRTree			-5.66%	-32.89%	-84.55%	-26.39%	-42.73%

Таблица 8: Результаты тестирования алгоритма RFRTree с другими известными алгоритмами (качество работы оценивается функционалом качества RMSE)

6 Заключение

В данной работе рассматривается задача восстановления регрессии. Для её решения применяются алгоритмы, основанные на построении полных регрессионных решающих деревьев (ПРРД). Ранее этот вид регрессионных деревьев был исследован другими авторами на задачах классификации по прецедентам и показал повышение качества решения по сравнению с наиболее известными методами синтеза регрессионных деревьев.

Разработаны алгоритмы NBFRTree, RFRTree и DFRTree синтеза ПРРД. Алгоритм NBFRTree предназначен для задач с целочисленной информацией. В алгоритме NBFRTree при построении регрессионного дерева на каждой итерации строится так называемая полная вершина, которой соответствует набор признаков, удовлетворяющих критерию ветвления в равной или почти равной мере. Ветвление в том алгоритме производится по каждому значению каждого признака из этого набора. Алгоритмы RFRTree и DFRTree предназначены для обработки вещественнозначной информации. В алгоритмах RFRTree и DFRTree при построении регрессионного дерева на каждой итерации строится так называемая полная вершина, которой соответствует набор пар $Z = \{(x_{i_1}, D(x_{i_1})), \ldots, (x_{i_t}, D(x_{i_t}))\}$, где $D(x_{i_j}), i_j \in \{1, ..., n\}, j \in \{1, ..., t\}$ — множество порогов для признака x_{i_j} . В этом наборе каждая пара $(x_{i_j}, D(x_{i_j}))$ удовлетворяет критерию ветвления в равной или почти равной мере. Затем для каждой такой пары строится «простая» внутренняя вершина x_{i_j}, из которой осуществляется ветвление с числом дуг, определяемым мощностью $D(x_{i_j})$. В обоих алгоритмах используется статистический критерий ветвления.

Алгоритм RFRTree строит ПРРД, в котором множество порогов $D(x_{i_j}), j \in \{1, \ldots, t\}$, это наилучший вариант случайного выбора порогов. Алгоритм DFRTree выбирает множество порогов $D(x_{i_j})$ по принципу равномерного распределения обучающих объектов в интервальном разбиении. По сравнению с классической конструкцией бинарного регрессионного дерева, регрессионные деревья в построенных алгоритмах позволяют более полно использовать имеющуюся информацию, при этом описание распознаваемого объекта может порождаться не одной ветвью, как в классическом решающем дереве, а несколькими ветвями.

На большом числе реальных задач проведено тестирование алгоритмов NBFRTree, RFRTree и DFRTree. Так как алгоритм NBFRTree предназначен для задач с целочисленной информацией, поэтому в задачах, в которых присутствовали признаки, принимающие вещественнозначные значения, была применена процедура перекодирования веще-

ственнозначных значений признака в целочисленные невысокой значности. Показано, что алгоритм RFRTree лучше алгоритма DFRTree в среднем на 2.5% по функционалу MAE и на 0.36% по функционалу RMSE, а также лучше алгоритма NBFRTree в среднем на 7.56% по функционалу MAE и на 1.91% по функционалу RMSE. Алгоритмы RFRTree, DFRTree и NBFRTree сравнивались также с другими алгоритмами синтеза регрессионных деревьев такими, как Random Forest, Decision Stump, REPTree и CART. Из экспериментов следует, что качество алгоритма RFRTree на большинстве задач выше качества алгоритмов Decision Stump, REPTree, M5P и CART в среднем на 68.3% по функционалу MAE и на 46.6% по функционалу RMSE. Кроме того на 11 из 15 задач алгоритм RFRTree показал результаты лучше, чем Random Forest в среднем на 45.36% по функционалу MAE и на 17.04% по функционалу RMSE.

Для заполнения пропусков в данных использовалось такие числовые характеристики множества значений признака как среднее арифметическое, медиана и мода. Указанные числовые характеристики вычислялись с учётом всех обучающих объектов и с учётом только тех из них, которые являются «ближайшими соседями». Пропуски в исходных признаковых описаниях объектов генерировались случайным образом. Рассматривались случаи с различным числом объектов, в описаниях которых встречаются пропуски. Наилучшие результаты получены при замене пропуска средним значением признака по всем обучающим объектам и при замене пропуска медианой тех значений признака, которые встречаются у ближайших соседей.

Таким образом, разработанные алгоритмы синтеза ПРРД для задач с вещественнозначными признаками могут быть успешно применены наравне с другими современными подходами к построению регрессионных деревьев.

Список литературы

[1] *Генрихов И. Е., Дюкова Е. В.* Классификация на основе полных решающих деревьев. — Ж. вычисл. матем. и матем. физ., 2012, том 52, номер 4, 750–761 с.

[2] *Генрихов И. Е., Дюкова Е. В., Журавлёв В.И.* О полных регрессионных решающих деревьях. — Машинное обучение и анализ данных, 2016. том 2, номер 1, 116–126 с.

[3] *Генрихов И. Е., Дюкова Е. В., Журавлёв В.И.* О полных регрессионных решающих деревьях, Междунар. конф. Интеллектуализация обработки информации-11. – М.: МАКС Пресс, 2016. 1–2 с.

[4] *Генрихов И. Е., Дюкова Е. В., Журавлёв В.И.* О полных решающих деревьях в задаче восстановления регрессии, Всеросс. конф. Математические методы распознавания образов-18. – М.: МАКС Пресс, 2017. 21–22 с.

[5] *Генрихов И. Е., Дюкова Е. В., Журавлёв В.И.* Построение и исследование полных решающих деревьев для задачи восстановления регрессии в случае вещественнозначной информации. — Машинное обучение и анализ данных, 2017. том 3, номер 2, 107–118 с.

[6] *Ai W. I., Langley P.* Induction of one-level decision trees // Proceedings of the Ninth International Conference on Machine Learning. — 1992. P. 233–240.

[7] *Breiman L., Friedman J.H., Olshen R.A., and Stone C.J.*. Classification and Regression Trees. — 1984.

[8] *Breiman L., Schapire E.* Random forests // Machine Learning. — 2001. P. 5–32.

[9] *Breiman, L.* Consistency for a simple model of Random Forests // Technical report, University of California at Berkeley, 2004.

[10] *Clifton D.* Classification and Regression Trees, Bagging, and Boosting Handbook of Statistics, 2005. Vol. 24. Pp. 303-329.

[11] *D. Li, J. Deogun, W. Spaulding, B. Shuart* Towards missing data imputation: A study of fuzzy k-means clustering method // Rough Sets and Current Trends in Computing. 2004. Vol. 3066. Pp. 573–579.

[12] *Djukova E.V., Peskov N.V.* A classification algorithm based on the complete decision tree. — Pattern Recognition and Image Analysis, 2007, Vol. 17, Issue 3, p. 363–367 363–367 p.

[13] *Elomaa T., Kaariainen M.* An analysis of reduced error pruning // Journal of Artificial Intelligence Research, 2001. Vol. 15. P. 163–187.

[14] *Genrikhov I.E., Djukova E.V., Zhuravlev V.I.* On Full Regression Decision Trees. — Pattern Recognition and Image Analysis. 2017, V. 27, Number 1, P. 1–7.c.

[15] *Gupta A., Lam M.* The weight decay backpropagation for generalizations with missing values // Annals of Operations Research. 1998. Vol. 78. Pp. 165–187.

[16] *Guvenir H., Uysal I.* An overview of regression techniques for knowledge discovery The Knowledge Engineering Review, 1999. Vol. 14:4. Pp. 319-340.

[17] *Kuncheva L. I.* Combining pattern classifiers methods and algorithms John Wiley and Sons, Inc., Hoboken, New Jersey, 2004. Pp. 154-163.

[18] *Liaw A., Wiener M.* Classification and Regression by randomForest // R News. 2002. Vol. 2, N 3. Pp. 18–22.

[19] *Lichman M.* Uci machine learning repository. — 2013. `http://archive.ics.uci.edu/ml`.

[20] *M. Zontul, F. Aydin, G. Dogan, S. Sener, O. Kaynar* Wind speed forecasting using reptree and bagging methods in kirklareli-turkey Journal of Theoretical and Applied Information Technology 10th October, 2013 Vol. 56 No.1

[21] *Quinlan J. R.* Learning with continuous classes. — 1992. P. 343–348.

[22] *Quinlan, J.R.* Induction of decision trees Machine Learning, 1986 Pp. 81-106.

[23] *Quinlan, J.R.* Combining instance-based and model-based learning Proc. ML'93 (ed P.E. Utgoff), San Mateo: Morgan Kaufmann, 1993

[24] *R. Timofeev* Classi?cation and Regression Trees (CART) Theory and Applications, 2004

[25] *Wang ,Y., Witten, I. H.* Induction of model trees for predicting continuous classes Poster papers of the 9th European Conference on Machine Learning, 1997

[26] *Wei-Yin Loh* Logistic Regression Tree Analysis Handbook of Engineering Statistics, H. Pham, ed., Springer, 2006 Pp. 537–549

[27] *Y. Yohannes, J. Hoddinott* Classification and regression trees: an introduction International Food Policy Research Institute 2033 K Street, N.W. Washington, D.C., 1999

[28] *Y. Zhao, Y. Zhan* Comparison of decision tree methods for ?nding active objects, 2007