Построение и исследование полных решающих деревьев для задачи восстановления регрессии в случае вещественнозначной информации*

И. Е. Генрихов[1]***, Е. В. Дюкова***[2]***, В. И. Журавлёв***[3]
ingvar1485@rambler.ru; edjukova@mail.ru; vadim091294@gmail.com
[1]ООО «Мобайл парк ИТ», Россия, г. Химки, ул. Панфилова, 21/1
[2]ФИЦ «Информатика и управление» РАН, Россия, г. Москва, ул. Вавилова, 44/2
[3]МГУ им. М. В. Ломоносова, Россия, г. Москва, Ленинские горы, 1

Рассматривается одна из актуальных задач машинного обучения — задача восстановления регрессии. Среди существующих подходов к решению этой задачи выделяют подход, основанный на построении регрессионных решающих деревьев (РРД). В данной работе рассматриваемая задача решается на основе построения так называемых полных k-арных РРД. По сравнению с классическим РРД конструкция полного РРД (ПРРД) позволяет более существенно использовать имеющуюся информацию, поскольку на каждой итерации в ветвлении участвуют все признаки, удовлетворяющие выбранному критерию. Ранее подход к построению ПРРД был исследован авторами на задачах восстановления регрессии с целочисленной информацией и показал повышение качества решения по сравнению с рядом других методов синтеза регрессионных деревьев. Получены новые результаты, связанные с построением полных k-арных РРД для задачи восстановления регрессии в случае вещественнозначной информации. Как известно, данный вид информации наиболее часто встречается на практике.

Ключевые слова: *задача восстановления регрессии; регрессионные деревья; полное решающее дерево*

DOI: 10.21469/22233792.3.2.02

1 Введение

Одной из центральных задач машинного обучения является задача восстановления регрессии. Рассматривается постановка этой задачи для случая вещественнозначной информации.

Исследуется множество объектов M. Объекты из M описываются системой признаков $\{x_1, \ldots, x_n\}$, где каждый признак $x_i, i \in \{1, ..., n\}$, — это результат измерения некоторой характеристики объекта. 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» $Y \subseteq \mathbb{R}$ 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 $A_T : M \longrightarrow Y$, ставящий в соответствие каждому объекту S из M значение y из Y.

Среди существующих подходов к решению задачи восстановления регрессии выделяют подход, основанный на построении РРД.

Процедура построения классического РРД представляет собой итерационный процесс. Как правило, на каждой итерации построения внутренней вершины дерева проводится

*Работа выполнена при финансовой поддержке РФФИ, проект № 16-01-00445.

разбиение текущих значений признака только по одному порогу. Иногда проводится разбиение по нескольким порогам (интервальное разбиение), при этом поиск оптимальных интервалов разбиения — это сложная в вычислительном плане задача. При синтезе таких деревьев на каждом шаге выбирается только один признак x и соответствующее ему множество порогов $D(x)$, удовлетворяющее выбранному критерию ветвления, и по нему осуществляется ветвление. Однако, если при построении дерева несколько различных пар $(x, D(x))$ удовлетворяют критерию ветвления в равной или почти равной мере, то среди них выбирается только одна пара (фактически случайным образом). При этом в зависимости от выбранного признака x и множества порогов $D(x)$ построенные деревья могут существенно отличаться как по составу используемых признаков, так и по своим распознающим качествам. Указанного недостатка лишена модель ПРРД [1, 2]. Идея полного решающего дерева впервые предложена в [3, 4] для задачи классификации по прецедентам. В ПРРД на каждой итерации строится так называемая полная вершина, которой соответствует набор $Z = \{(x_{i_1}, D(x_{i_1}), \ldots, (x_{i_t}, D(x_{i_t}))\}, i_j \in \{1, \ldots, n\}, j \in \{1, \ldots, t\}$, в котором каждый признак x_{i_j} и соответствующее ему множество порогов $D(x_{i_j})$ удовлетворяют критерию ветвления в равной или почти равной мере. Затем для каждой пары $(x_{i_j}, D(x_{i_j})) \in Z$ строится внутренняя (простая) вершина, из которой осуществляется ветвление. В [1, 2] ПРРД были построены для задачи восстановления регрессии в случае целочисленной информации.

Основной целью данной работы является конструирование и исследование ПРРД для задач с вещественнозначными данными. В работе построены алгоритмы DFRTree (Defined Full Regression Tree) и RFRTree (Random Full Regression Tree), строящие полные k-арные регрессионные деревья, где k — максимальное число ребер, выходящих из простой вершины дерева. При построении очередной простой вершины в алгоритмах DFRTree и RFRTree из множества различных значений признака выбирается $k-1$ пороговых значений, которые образуют k интервалов. В обоих алгоритмах используется критерий ветвления, являющийся модификацией критерия ветвления алгоритма CART (classification and regression tree) на случай k-арного дерева. Алгоритмы DFRTree и RFRTree различаются способом выбора пороговых значений. Проведено тестирование алгоритмов RFRTree и DFRTree на реальных задачах. Показано, что на большинстве рассмотренных в работе задач алгоритм RFRTree работает лучше других алгоритмов восстановления регрессии, участвовавших в тестировании, среди которых алгоритмы Random Forest [5], REPTree [6], M5P [7], CART [8], DS (Decision Stump) [9], NBFRTree (nonbinary full regression tree) и DFRTree [1, 2].

2 Основные понятия и полученные ранее результаты

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

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

На первом шаге $\breve{T} = T, X(\breve{T}) = \{x_1, \ldots, x_n\}$. 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-ю подвыборку попадают объекты из $\breve{T}$, для которых значение 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 $x \in X(\breve{T})$.

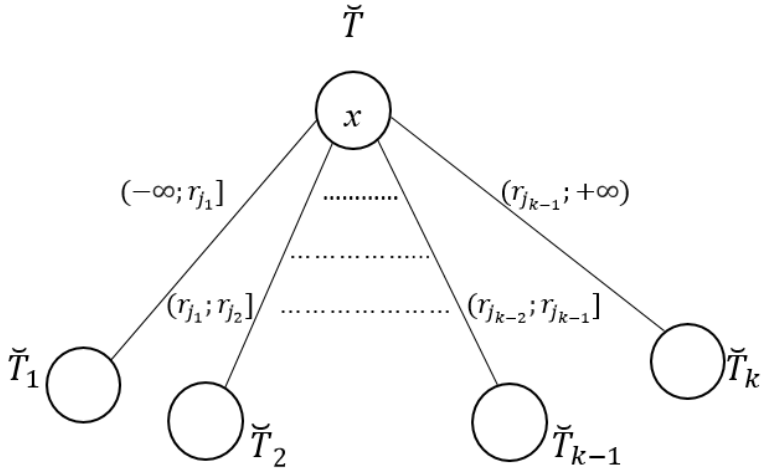

Рис. 1 Пример ветвления из вершины x в k-арном РРД

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 критерий ветвления алгоритма CART, обобщенный на случай k-арного дерева.

Пусть $\breve{T}_i = S_1^i, \ldots, S_{u_i}^i, y_t^i, i \in \{0, 1, \ldots, m\}$ — 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 $r_{j_1}, \ldots, r_{j_{k-1}}$. Таким образом, значения признака x разбиваются на k полуинтервалов $\{(-\infty, r_{j_1}], (r_{j_1}, r_{j_2}], \ldots, (r_{j_{k-1}}, +\infty)\}$. Обучающая выборка $\breve{T}_i$ разбивается по этому признаку на k подвыборок $\breve{T}_{i_1}, \ldots, \breve{T}_{i_k}$. Вычисляются величины:

$$\bar{y}_{\breve{T}_i} = \frac{1}{u_i} \sum_{t=1}^{u_i} y_t^i;$$

$$\text{Variance} = \frac{1}{u_i} \sum_{t=1}^{u_i} (y_t^i)^2 - \Big[\frac{1}{u_i} \sum_{t=1}^{u_i} (y_t^i)\Big]^2;$$

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

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

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. 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, REPTree и Decision Stump. 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.). В алгоритме Random Forest используется энтропийный критерий ветвления и процедура «бэггинг». Процедура бэггинга над БРРД заключается в использовании композиции БРРД, каждое из которых строится независимо. Для построения очередного дерева композиции случайным образом выбирается (с возвращением) некоторое подмножество обучающих объектов из исходной выборки. Результат распознавания определяется путем усреднения значений целевой переменной по всем построенным БРРД. Таким образом, деревья компенсируют ошибки друг друга.

Ранее в [1, 2] автором был реализован алгоритм NBFRTree, строящий k-арное ПРРД для задач с целочисленной информацией, где k — максимальное число ребер, выходящих из простых вершин дерева. В алгоритме NBFRTree используется критерий ветвления, являющийся модификацией критерия ветвления алгоритма CART на случай k-арного дерева (описан выше). Показано, что идея ПРРД может быть успешно использована наравне с другими известными подходами к синтезу РРД для задач с целочисленной информацией.

В следующем разделе описываются алгоритмы RFRTree и DFRTree, строящие ПРРД для задач с вещественнозначной информацией.

3 Алгоритмы построения полных регрессионных решающих деревьев DFRTree и RFRTree

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.

Пусть далее g — число различных значений признака x. В алгоритме DFRTree из множества значений признака x выбирается $k-1, 2 \leqslant k \leqslant g$, 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.

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

Пусть $\{z_1, z_2, \ldots, z_{k-1}\}, k \geqslant 2$, — 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 $\{(-\infty, z_1], (z_1, z_2], \ldots, (z_{k-1}, +\infty)\}$.

Пусть σ — метка одной из дуг, выходящих из вершины $x, \sigma \in \{(-\infty, z_1], (z_1, z_2], \ldots$ $\ldots, (z_{k-1}, +\infty)\}$. 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 раздела.

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 $x_{j_1}^{\sigma_1} \cdots x_{j_r}^{\sigma_r}$. 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. 2 приведен 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 $(B_i, w(v_i))$, $i = 1, 2, \ldots, l, l \geqslant 1$. Для каждой висячей вершины v_i осуществляется проверка принадлежности описания объекта S интервалу истинности N_{B_i}.

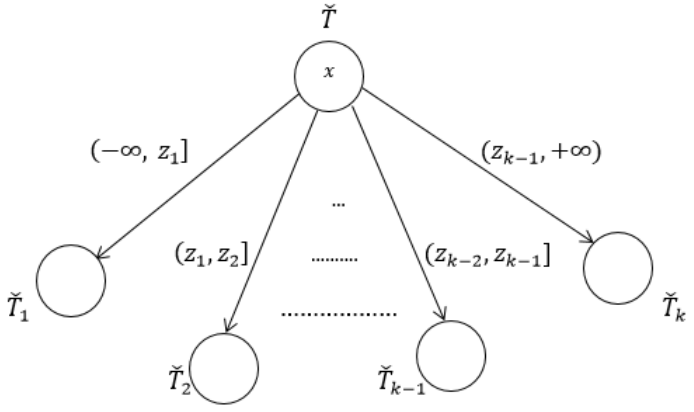

Рис. 2 Пример ветвления из простой вершины x в алгоритмах DFRTree и RFRTree

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

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

Опишем критерий ветвления, используемый в алгоритмах DFRTree и RFRTree.

Пусть $\breve{T}_i = S_1^i, \ldots, S_{u_i}^i, y_t^i, i \in \{0, 1, \ldots, m\}$ — 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 $\breve{T}_{i_1}, \ldots, \breve{T}_{i_k}$. Вычисляются величины:

$$\mathrm{SE}\,(x,k) = \frac{1}{u_i}\left\{ \sum_{S_t^i \in \breve{T}_{i_1}} (y_t^i - \bar{y}_{\breve{T}_{i_1}})^2 + \cdots + \sum_{S_t^i \in \breve{T}_{i_k}} (y_t^i - \bar{y}_{\breve{T}_{i_k}})^2 \right\};$$

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

Пусть $C_{\min} = \min\limits_{x \in X(\breve{T})} C(k,x)$ и $C_{\max} = \max\limits_{x \in X(\breve{T})} C(k,x)$. Сначала для каждого признака $x \in X(\breve{T}_i)$ вычисляется величина $C(k,x) = \mathrm{Variance} - \mathrm{SE}\,(x,k)$. 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 $0{,}7 \leqslant C^*(k,x) \leqslant 1$. В случае, когда $C_{\max} = C_{\min}$, разбиение производится по всем признакам из $X(\breve{T}_i)$.

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

4 Тестирование алгоритмов DFRTree и RFRTree

Алгоритмы были протестированы на 15 реальных задачах из ресурса UCI [10]. 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.

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

Таблица 1 Оптимальное значение ε

Задача	ε
Data1	0,01
Data2	1000
Data3	0,1
Data4	1
Data5	0,01
Data6	0,5
Data7	0,1
Data8	1
Data9	1
Data10	80
Data11	170
Data12	0
Data13	1
Data14	1
Data15	4

Для оценки качества работы алгоритмов была применена кросс-валидация по k фолдам. Исходные данные разбивались на k подвыборок, $k \geqslant 2$. 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:

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

где y_i — значения целевых переменных, а f_i — значения, выданные алгоритмом.

Для алгоритма RFRTree был дополнительно исследован параметр h, описанный в разд. 3, — число выполнений процедуры выбора пороговых значений. В табл. 2 приведены результаты тестирования при $h = 1, 5, 10, 20, 50$, $C^* = 0{,}85$ и $k = 6$.

Как видно из табл. 2, на большинстве задач наилучшие результаты получились при $h = 10$, в том числе и по процентному отклонению от $h = 1$.

В табл. 3 представлены результаты тестирования алгоритмов RFRTree, DFRTree и NBFRTree [1, 2]. 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, поэтому в задачах, в которых присутствовали признаки, принимающие вещественнозначные значения, была применена процедура перекодирования вещественнозначных значений признака в целочисленные. Производилась она следующим образом. Пусть $\{c_1, \ldots, c_u\}$ — множество различных значений признака x, $c_{i+1} > c_i, 1 \leqslant i \leqslant u-1$. Выбирается t порогов для признака x, делящих обучающую выборку по этому признаку на t равных частей. Для однообразия эксперимента t положили равным 5.

Таблица 2 Результаты тестирования алгоритма RFRTree при различных значениях параметра h ($k = 6$)

Задача	Размер $m \times n$	$h = 1$		$h = 5$		$h = 10$		$h = 20$		$h = 50$	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Data1	100 × 10	0,210	0,372	0,182	0,364	**0,173**	**0,338**	0,184	0,373	0,201	0,409
Data2	160 × 25	1296,583	1993,421	1280,348	**1916,560**	**1240,264**	1928,312	1253,729	1957,921	1280,794	1918,816
Data3	167 × 4	0,401	0,923	0,265	0,507	0,265	0,535	0,272	0,531	**0,253**	**0,481**
Data4	198 × 33	0,298	0,546	0,288	0,537	**0,268**	**0,517**	0,295	0,542	**0,268**	**0,517**
Data5	308 × 6	4,341	8,184	3,335	5,737	2,611	4,617	2.400	3.960	**2,164**	**3,395**
Data6	392 × 8	1,997	2,823	**1,989**	**2,775**	2.057	2.811	2,079	2,935	2,103	2,907
Data7	506 × 13	3,312	4,816	2,934	4,319	**2,700**	**3,885**	2,746	4,027	2,836	3,933
Data8	517 × 12	20,911	54,588	19,567	53,834	**18,014**	**48,460**	19,940	53,923	20,300	57,153
Data9	569 × 30	**0,036**	**0,181**	0,040	0,172	0,049	0,214	0,045	0,193	0,049	0,211
Data10	1059 × 68	13,615	17,021	13,562	17,001	**13,447**	**16,944**	13,759	17,120	13,568	17,022
Data11	1059 × 68	35,309	**44,534**	**35,271**	44,745	35,400	45,322	35,466	45,832	35,962	46,340
Data12	1503 × 5	2,662	3,435	2,534	3,291	**2,370**	**3,125**	2,396	3,143	2,412	3,184
Data13	1599 × 11	0,393	0,700	0,387	0,689	0,384	0,699	**0,371**	**0,674**	0,372	0,679
Data14	4898 × 11	0,388	0,712	0,385	0,708	**0,384**	**0,701**	0,390	0,708	0,396	0,718
Data15	9568 × 4	3,421	4,482	3,135	4,122	3,112	4,129	3,111	4,122	**3,075**	**4,101**
Отклонение от $h = 1$				+6,54%	+7,67%	+8,22%	+8,89%	+7,26%	+8,04%	+6,65%	+7,57%

Таблица 3 Результаты тестирования алгоритмов RFRTree, DFRTree и NBFRTree

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%

Для большинства рассмотренных задач наилучшие результаты достигаются при значении $C^* = 0{,}7$.

Для всех задач применялась кросс-валидация по 10 фолдам. Для большей надежности эксперимента кросс-валидация по 10 фолдам производилась 10 раз, после каждой итерации выборка перемешивалась.

В табл. 4 и 5 приведены результаты тестирования алгоритма RFRTree 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. Из этих таблиц видно, что на 11 из 15 реальных задач с функционалом качества MAE наилучшие результаты показал алгоритм RFRTree, на трех — Random forest

Таблица 4 Результаты тестирования алгоритма RFRTree с другими известными алгоритмами (качество работы оценивается функционалом качества MAE)

Задача	Размер $m \times n$	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 5 Результаты тестирования алгоритма RFRTree с другими известными алгоритмами (качество работы оценивается функционалом качества RMSE)

Задача	Размер $m \times n$	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%

и на одной — CART. На 10 из 15 реальных задач с функционалом качества RMSE наилучшие результаты показал алгоритм RFRTree, на трех — Random Forest, на одной — алгоритмы CART и DS.

5 Заключение

В данной работе рассматривается задача восстановления регрессии с вещественнозначными признаками. Для ее решения применяются алгоритмы, основанные на построении ПРРД. Ранее этот вид регрессионных деревьев был исследован авторами на задачах с целочисленными признаками и показал повышение качества решения по сравнению с наиболее известными методами синтеза регрессионных деревьев.

Разработаны алгоритмы RFRTree и DFRTree синтеза ПРРД. В этих алгоритмах 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 $D(x_{i_j}), i_j \in \{1, \ldots, n\}, j \in \{1, \ldots, t\}$, — 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 RFRTree и DFRTree. Показано. что алгоритм RFRTree лучше алгоритма DFRTree в среднем на 2,5% по функционалу MAE и на 0,36% по функционалу RMSE. Алгоритмы RFRTree и DFRTree сравнивались также с другими алгоритмами синтеза регрессионных деревьев, такими как Random Forest, DS, REPTree и CART. Из экспериментов следует, что качество алгоритма RFRTree на большинстве задач выше качества алгоритмов DS, 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.

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

Литература

[1] *Генрихов И. Е., Дюкова Е. В., Журавлёв В. И.* О полных регрессионных решающих деревьях // Машинное обучение и анализ данных, 2016. Т. 2. № 1. С. 116–126.

[2] *Genrikhov I. E., Djukova E. V., Zhuravlev V. I.* On full regression decision trees // Pattern Recognition Image Anal., 2017. Vol. 27. No. 1. P. 1–7.

[3] *Djukova E. V., Peskov N. V.* A classification algorithm based on the complete decision tree // Pattern Recognition Image Anal., 2007. Vol. 17. No. 3. P. 363–367.

[4] *Генрихов И. Е., Дюкова Е. В.* Классификация на основе полных решающих деревьев // Ж. вычисл. матем. матем. физ., 2012. Т. 52. № 4. С. 750–761.

[5] *Breiman L.* Random forests // Machine Learning, 2001. Vol. 45. No. 1. P. 5–32.

[6] *Elomaa T., Kaariainen M.* An analysis of reduced error pruning // J. Artif. Intell. Res., 2001. Vol. 15. P. 163–187.

[7] *Quinlan J. R.* Learning with continuous classes // Australian National Conference on Artificial Intelligence Proceedings. — Singapore: World Scientific, 1992. P. 343–348.

[8] *Breiman L., Friedman J. H., Olshen R. A., Stone C. J.* Classification and regression trees. — CRC Press, 1984. 368 p.

[9] *Iba W., Langley P.* Induction of one-level decision trees // 9th Conference (International) on Machine Learning Proceedings. — San Francisco, CA, USA: Morgan Kaufmann, 1992. P. 233–240.

[10] *Lichman, M.* UCI Machine Learning Repository. // Irvine, CA, USA: University of California, School of Information and Computer Science, 2013. http://archive.ics.uci.edu/ml.

Поступила в редакцию 30.08.2017

Construction and investigation of full regression trees in regression restoration problem in the case of real-valued information*

I. E. Genrikhov, E. V. Djukova, and V. I. Zhuravlyov

ingvar1485@rambler.ru; edjukova@mail.ru; vadim091294@gmail.com

[1]LLC Mobile park IT, 21/1 Panfilova Str., Khimki, Moscow region, Russia

[2]Federal Research Center "Computer Science and Control" of RAS, 44/2 Vavilova Str., Moscow, Russia

[3]Lomonosov Moscow State University, 1 Leninskie Gory, Moscow, Russia

Background. The regression restoration problem with real data is considered. Typically, this type of information is most often encountered in practice (for example, in problems of medical diagnosis or banking scoring). The approach based on the construction of regression trees is highlighted among the existing approaches. The most known among algorithms of regression trees synthesis (for example, algorithms CART (classification and regression tree) and Random Forest) are based on use of the elementary trees, namely, binary regression trees. As a rule, in this case, in the synthesis of regression trees, the current values of the feature are splitted by only one threshold at each step of constructing the inner node of the tree. Sometimes, a splitting into several thresholds is performed (interval splitting), while searching for optimal splitting intervals computationally is a complex task. During the synthesis of such trees, only one feature and the corresponding set of thresholds are selected at each step, which satisfies the selected branch criterion, and branching is performed based on it. However, if several different pairs (a feature – a set of transcoding thresholds) satisfy the branching criterion in equal or almost equal measure when building a tree, then only one of them (in fact, randomly) is selected. Thus, depending on the selected feature and the set of thresholds, the constructed trees can differ significantly, both in the composition of the features used and in their recognizing qualities.

Methods. An approach to the construction of regression trees based on the construction of the so-called full decision tree is applied. In addition, various ways of selecting a set of thresholds for feature at each step of tree synthesis have been investigated. Previously, this approach was investigated only on regression restoration problem with integer data and showed an improvement in the quality of the solution in comparison with the known methods of synthesis of regression trees. In a full regression tree, a so-called full node on each iteration for a problem with real features is constructed. A set of pairs of feature–threshold transcoding corresponds

*The research was supported by the Russian Foundation for Basic Research (grant 16-01-00445).

to it, in which each pair of feature–threshold satisfies the selected branching criterion. Further, a simple inner node is constructed for each pair from this set, from which branching is performed. Compared to the classical construction and the standard method of selecting only one threshold for the feature, full regression tree allows fuller use of the available information, while the description of the recognized object can be generated not only by one branch as in a classical tree, but by several branches.

Results. Two synthesis algorithms of regression trees — DFRTree (defined full regression tree) and RFRTree (random full regression tree) — are developed. The RFRTree algorithm constructs a full regression tree in which the best set of thresholds for the feature is selected at each step of the synthesis by using a statistical criterion of estimating various random partitions. The DFRTree algorithm also constructs a full regression tree, but selects the set of thresholds for the feature for which approximately the same number of training objects fall into the resulting intervals. It is shown that the best results were obtained using the RFRTree algorithm. A comparison of 15 real problems of RFRTree and DFRTree algorithms with known regression trees synthesis algorithms, such as the Random Forest, Decision Stump, REPTree, and CART, is carried out. It is shown that quality of the RFRTree algorithm is higher than the quality of the Decision Stump, REPTree, and CART algorithms, and it is as good as the Random Forest algorithm and, in some cases, shows the best results.

Concluding Remarks. It is shown that the developed algorithms for the synthesis of full regression trees for solving the regression restoration problem with real data are not inferior to the known algorithms for the synthesis of regression trees and can be successfully applied on a par with other modern approaches to constructing regression trees.

Keywords: *regression restoration problem; regression trees; full decision tree*

DOI: 10.21469/22233792.3.2.02

References

[1] Genrikhov, I. E., E. V. Djukova, and V. I. Zhuravlyov. 2016. O polnykh regressionnykh reshayushikh derevyakh [On complete regression and decisive trees]. *Mashinnoe obuchenie i analiz dannyh* [Machine Learning Data Anal.] 2(1):116–126.

[2] Genrikhov, I. E., E. V. Djukova, and V. I. Zhuravlev. 2017. On full regression decision trees. *Pattern Recognition Image Anal.* 27(1):1—7.

[3] Djukova, E. V., and N. V. Peskov. 2007. A classification algorithm based on the complete decision tree. *Pattern Recognition Image Anal.* 17(3):363–367.

[4] Genrikhov, I. E., and E. V. Djukova. 2012. Classification based on full decision trees. *Comp. Math. Math. Phys.* 52(4):653–663.

[5] Breiman, L. 2001. Random forests. *Machine Learning* 45(1):5–32.

[6] Elomaa, T., and M. Kaariainen. 2001. An analysis of reduced error pruning. *J. Artif. Intel. Res.* 15:163–187.

[7] Quinlan, J. R. 1992. Learning with continuous classes. *Australian National Conference on Artificial Intelligence Proceedings*. Singapore: World Scientific. 343—348.

[8] Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and regression trees*. CRC Press. 368 p.

[9] Iba, W., and P. Langley. 1992. Induction of one-level decision trees. // *9th Conference (International) on Machine Learning Proceedings*. San Francisco, CA: Morgan Kaufmann. 233–240.

[10] Lichman, M. UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. Available at: http://archive.ics.uci.edu/ml (accessed November 29, 2017).

Received August 30, 2017