Реализация и сравнение алгоритмов построения деревьев решений для задач классификации объектов

В.П. Клюев[1], А.В. Куприянов[1,2]

[1]Самарский национальный исследовательский университет им. академика С.П. Королева, Московское шоссе 34А, Самара, Россия, 443086
[2]Институт систем обработки изображений РАН – филиал ФНИЦ «Кристаллография и фотоника» РАН, Молодогвардейская 151, Самара, Россия, 443001

Аннотация. Статья посвящена решению задачи классификации объектов, с помощью метода деревьев решений, на языке программирования Python. В статье рассматриваются два алгоритма для построения деревьев решений. Проведены экспериментальные исследования работы алгоритмов над тестовыми данными и проанализированы полученные результаты. Сделаны выводы о положительных и отрицательных сторонах рассмотренных алгоритмов и метода деревьев решений.

1. Введение

Стремительное развитие информационных технологий, в частности, прогресс в методах сбора, хранения и обработки данных позволил собирать огромные массивы данных, которые необходимо анализировать. Объемы этих данных настолько велики, что возможностей экспертов уже не хватает. Это породило спрос на методы автоматического анализа данных, который с каждым годом постоянно увеличивается. Деревья решений – один из методов автоматического анализа данных. Область применения деревьев решений в настоящее время очень широка. Деревья решений являются прекрасным инструментом в системах поддержки принятия решений, интеллектуального анализа данных. В состав многих пакетов, предназначенных для интеллектуального анализа данных, уже включены методы построения деревьев решений. В областях, где высока цена ошибки, они послужат отличным подспорьем [1]. Деревья решений успешно применяются для решения практических задач в следующих областях:

- Банковское дело.
- Промышленность.
- Медицина.
- Молекулярная биология.

Все задачи, решаемые деревьями решений, можно разделить на три типа [2]:

- Классификация данных.
- Описание данных.
- Регрессия.

Дерево решений – это древовидный граф, состоящий из узлов и листьев, соединенных между собой дугами. В узлах графа происходит принятие решений, а листья указывают на классы. Граф дерева решений должен быть ациклический, иначе он перестает быть

древовидным и его использование для принятия решений вызовет большие трудности [1]. В данный момент существует большое количество алгоритмов, которые могут построить дерево решений: ID3, CART, C4.5, NewId, ITrule, CHAID, CN2 [2] и т.д. Самыми популярными на сегодняшний день являются алгоритмы: ID3 [3], C4.5 [3, 4] и CART [5].

В данной статье рассмотрена реализация алгоритмов ID3 и CARD на языке программирования Python. Проведено экспериментальное исследование алгоритмов и проанализированы полученные результаты.

2. Построение дерева решений

Имеем обучающее множество T. Оно состоит из объектов, у которых есть m атрибутов, характеризующих объект. По одному или нескольким атрибутам можно однозначно сказать, что объект принадлежит некоторому классу. Обозначим классы, к которым принадлежат объекты, как $C_1, C_2, ..., C_k$. Существуют три ситуации.

1. Множество T состоит из объектов принадлежащих одному классу C_k. Тогда T – это лист, определяющий класс C_k.
2. Множество T пустое. Тогда T - это снова лист, который принадлежит такому же классу, что и его родитель.
3. Множество T состоит из объектов, относящихся к разным классам. Множество T следует разбить на подмножества. Для этого выбирается один из атрибутов, имеющий различные значения $O_1, O_2, ..., O_m$. T разбивается на подмножества $T_1, T_2, ..., T_n$, в зависимости от значения атрибута O_i для выбранного признака. Это процедура будет рекурсивно продолжаться до тех пор, пока конечное множество не будет состоять из примеров, относящихся к одному классу.

3. Алгоритмы

Рассмотренные в данной статье алгоритмы построения дерева решений применяют различные критерии для выбора разбиения, но метод нахождения самих разбиений схож. A_i – числовой признак, то множество T разбивается на два подмножества. При этом необходимо выбрать некий порог разбиения, с которым будут сравниваться все значения признака. Обозначим множество значений признака A_i через $v = \{v_1, v_2, ..., v_n\}$. Сначала следует отсортировать значения по возрастанию. Тогда любое значение, лежащее между v_i и v_{i+1}, делит всё множество на два подмножества и в качестве порога можно выбрать среднее (1) между значениями v_i и v_{i+1}.

$$t_i = \frac{v_i + v_{i+1}}{2}. \quad (1)$$

Таким образом, имеется n-1 потенциальное пороговое значение для разбиения по атрибуту. Вычисления применяются ко всем возможным разбиениям, и в качестве рабочего выбирается наиболее информативное разбиение.

3.1. Алгоритм ID3

Данный алгоритм впервые был представлен Дж. Р. Куинланом в книге «Машинное обучение», опубликованной в 1992 году. Алгоритм ID3 работает рекурсивно, разбивая по выбранному атрибуту в каждом узле множество объектов на подмножества, начиная с корня дерева, в котором содержаться все объекты [3]. По каждому атрибуту A_i можно разбить множество T на подмножества $T_1, T_2, ..., T_n$. Пусть $N_{C_j}^T$ - количество объектов из множества T, принадлежащих классу C_j. Тогда вероятность того, что выбранный объект из множества T принадлежит классу C_j:

$$P_j = \frac{N_{C_j}^{T}}{|T|}. \quad (2)$$

Энтропия множества T будет имеет вид (3):

$$H(T) = -\sum_{j=1}^{k} P_j \log_2(P_j). \quad (3)$$

Условная энтропия (4) объектов T при рассматриваемом признаке, по которому было проведено разбиение X:

$$H(T|X) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} H(T_i). \quad (4)$$

Далее для каждого из признаков вычисляется количество информации (5):

$$I(X) = H(T) - H(T|X). \quad (5)$$

Выбирается наиболее информативное разбиение [4].

3.2. Алгоритм CART

Алгоритм CART (Classification and Regression Tree), как видно из названия, решает задачи классификации и регрессии. Он разработан в 1974-1984 годах профессорами Leo Breiman, Jerry Friedman, Charles Stone и Richard Olshen [5]. Алгоритм строит бинарные деревья, имеющие двух потомков в каждом узле дерева. На каждом шаге построения дерева правило, формируемое в узле, делит заданную обучающую выборку на две части - часть, в которой выполняется правило и часть, в которой правило не выполняется. Для выбора оптимального правила используется функция оценки качества разбиения. Если набор объектов T содержит данные k классов, тогда индекс Gini (6) определяется как:

$$\text{Gini}(T) = 1 - \sum_{j=1}^{k} p_j^2, \quad (6)$$

где p_j вычисляется по формуле (2). При разбиении множества объектов на два подмножества показатель качества разбиения (7) будет равен:

$$\text{Gini}_{\text{split}}(T_1, T_2) = \frac{|T_1|}{|T|}\text{Gini}(T_1) + \frac{|T_2|}{|T|}\text{Gini}(T_2). \quad (7)$$

Наилучшим считается разбиение, для которого индекс $\text{Gini}_{\text{split}}(T_1, T_2)$ минимален. Обозначив количество текстов i-го класса в левом и правом потомке через l_t и r_t оценить разбиении (8) можно по формуле:

$$\text{Gini}_{\text{split}}(T_1, T_2) = \frac{|T_1|}{|T|}\left(1 - \sum_{i=1}^{k}\left(\frac{l_t}{|T_1|}\right)^2\right) + \frac{|T_2|}{|T|}\left(1 - \sum_{i=1}^{k}\left(\frac{r_t}{|T_2|}\right)^2\right) \to \min. \quad (8)$$

Алгоритм последовательно сравнивает все возможные разбиения и выбирает наилучший признак и наилучшее разбиение для него [5-6].

4. Экспериментальные исследования

4.1 Тестовые данные

Алгоритмы были реализованы на языке программирования Python и протестированы на тестовых данных. В качестве тестовых данных использовались значения, полученные во время кластеризации изображения. Для каждого объекта был известен класс, к которому он относится. Размеры тренировочной и тестовой выборки для каждого класса приведены в таблице 1.

Таблица 1. Тестовые данные.

Классы	Тренировочная выборка	Тестовая выборка	Всего
C1	5492	5491	10983
C2	2902	2902	5804
C3	2987	2987	5974
C4	5766	5765	11531
C5	31543	31543	63086
C6	5556	5556	11112
C7	1537	1537	3074
C8	16584	16585	33169
C9	6796	6796	13592
C10	19230	19229	38459
Всего	98403	98401	196794

4.2 Исследование на двух классах

Вначале алгоритмы были протестированы на выборке из двух классов. Тестовые испытания алгоритмов на двух классах C1 и C2 показали, что алгоритмы выбирают одинаковые разбиения для построения дерева, а следовательно построили идентичные деревья. Модель дерева, построенного алгоритмами, приведена на рисунке 1. Видно, что классификация произошла за одну итерацию. Был выбран атрибут B013_L1GST с пороговым значением 10,5.

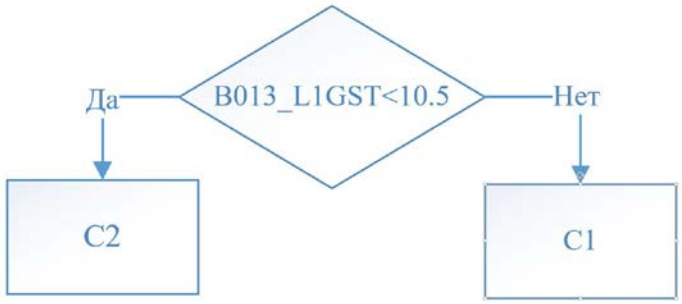

Рисунок 1. Дерево решений, построенное алгоритмом ID3 и CART на двух классах.

Дерево решений было протестировано на тестовой выборке. Результат тестирования приведен в таблице 2. Мы видим, что ошибка первого рода класса C1 значительно выше ошибки второго рода.

Таблица 2. Результат тестирования дерева решений, построенного алгоритмами ID3 и CART на двух классах.

Определенный класс	C1	C2
C1	5206	1
C2	285	2901
Не определен	0	0

Алгоритм ID3 справился с построением дерева за 3.8768 секунды, алгоритм CARD за 3.792.

4.3. Механизм урезания ветвей.

Был разработан механизм урезания ветвей, который применяется в экспериментальных исследованиях. Данный механизм нужен для уменьшения излишнего разрастания дерева. Если в узле дерева количество объектов некоторый класс (9) преобладает над другими классами с некоторой заданной погрешностью, то этот узел можно считать листом преобладающего класса. Для этого должно выполняться условие (10).

$$N_{\max}^{T} = \max(N_{C_j}^{T}); \quad (9)$$

$$\frac{N_{\max}^{T}}{N_{C_j}^{T}} < \varepsilon, \quad (10)$$

где ε – заданная точность построения.

4.4. Исследование на десяти классах

Тестирование алгоритмов на полной выборке ярко выявило различие алгоритмов. Результаты тестирования приведены в таблицах 3 и 4.

Таблица 3. Результат тестирования дерева решений, построенного алгоритмом ID3 на десяти классах.

Определенный класс	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	5487	0	0	5	0	0	0	0	0	0
C2	0	568	5	0	0	0	0	51	18	0
C3	0	0	10	0	0	0	0	0	0	0
C4	3	0	0	5760	0	0	0	1	0	0
C5	0	2	22	0	20349	5	0	10	0	0
C6	0	0	0	0	0	5555	0	0	0	0
C7	0	0	0	0	0	0	1534	0	0	0
C8	0	1	2	0	0	0	0	25	0	0
C9	1	9	0	0	0	1	3	0	6778	0
C10	0	54	1	0	0	0	0	0	0	19229
Не определен	0	2268	2947	0	11194	0	0	16498	0	0

Таблица 4. Результат тестирования дерева решений, построенного алгоритмом CARD на десяти классах.

Определенный класс	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	5440	0	0	0	0	0	0	0	0	0
C2	0	2897	0	0	2	0	0	0	0	0
C3	0	0	2987	0	0	0	0	0	0	0
C4	45	0	0	5765	0	0	0	0	0	0
C5	0	1	0	0	31541	0	0	0	0	0
C6	0	0	0	0	0	5556	0	0	0	0
C7	6	0	0	0	0	0	1537	0	0	0
C8	0	0	0	0	0	0	0	16585	0	0
C9	0	0	0	0	0	0	0	0	16585	0
C10	0	4	0	0	0	0	0	0	0	19229
Не определен	0	0	0	0	0	0	0	0	0	0

Из результатов видно, что алгоритм ID3 не смог справится с большим количеством классов и противоречивостью данных. Классы C2, C3 и C8 алгоритм не смог классифицировать. Ошибка классификации класса C5 составила 35%. Время построения дерева алгоритмом ID3 составило 1555.5893 секунды. Результаты работы ID3 сложно назвать приемлемыми.

У алгоритма CARD время построения дерева составило 124.6229 секунд. Есть небольшие ошибки классификации, но они объясняются возмущениями в тестовых данных.

5. Заключение

В статье были рассмотрены алгоритмы построения деревьев решений ID3 и CART. Рассмотрены критерии, которые используют эти алгоритмы для определения лучшего разбиения. Используя эти алгоритмы, были построены деревья решений сначала на выборке, состоящей из двух классов, а потом на полной выборке, состоящей из десяти классов. Построенные деревья были протестированы, и было выявлено, что на небольшом объеме оба алгоритма справляются со своими задачами и даже дают одинаковый результат. Был разработан механизм урезания ветвей, для предотвращения излишнего разрастания дерева. На полной выборке алгоритм ID3 не смог удовлетворительно классифицировать объекты и выходил из цикла построения по критерию прекращения, не определяя объект. Алгоритм CARD на полной выборке построил небольшое восьмиуровневое дерево и справился со своей задачей.

По результатам работы можно сделать вывод, что деревья решений справляются с задачами классификации, но большинство из известных алгоритмов построения являются "жадными алгоритмами". Это значит, что если один раз был выбран атрибут, и по нему было произведено разбиение на подмножества, то алгоритм не может вернуться назад и выбрать другой атрибут, который дал бы лучшее разбиение.

У алгоритма ID3 существует модификация, которая имеет название C4.5. В дальнейшей своей работе я планирую реализовать этот алгоритм и сравнить его с алгоритмом CARD. Для лучшего алгоритма планирую разработать многопоточную реализацию.

6. Благодарности

Работа выполнена при поддержке Федерального агентства научных организаций (соглашение № 007-ГЗ/Ч3363/26).

7. Литература

[1] Деревья классификации и регрессии [Электронный ресурс]. – Режим доступа: http://www.williamspublishing.com/PDF/978-5-8459-1170-4/part.pdf (29.09.2017).
[2] Методы классификации и прогнозирования. Деревья решений [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/department/database/datamining/9/1.html (12.10.2017).
[3] Quinlan R. C4.5: Programs for Machine Learning. – San Mateo, CA: Morgan Kaufmann, 1993. – 302 p.
[4] Деревья решений – C4.5 математический аппарат [Электронный ресурс]. – Режим доступа: http://www.basegroup.ru/library/analysis/tree/math_c45_part1/ (29.10.2017).
[5] Деревья решений – CART математический аппарат [Электронный ресурс]. – Режим доступа: http://www.basegroup.ru/library/analysis/tree/math_cart_part1/ (15.11.2017).
[6] Buntine, W. A theory of classification rules / W. Buntine, 1992.

Implementation and comparison of algorithms for building decision trees for the tasks of object classification

V.P. Klyuev[1], A.V. Kupriyanov[1,2]

[1]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086
[2]Image Processing Systems Institute of RAS - Branch of the FSRC "Crystallography and Photonics" RAS, Molodogvardejskaya street 151, Samara, Russia, 443001

Abstract. The article is devoted to solving problems of classifying objects using the method of decision trees in the programming language Python. The article discusses two algorithms for building decision trees. Experimental research of work of algorithms on test data and analyzed the results. The author has made conclusions about positive and negative aspects of the considered algorithms and the method of decision trees.

Keywords: Analysis, Classification, Decision trees, ID3, CARD.