Деревья решений — C4.5 математический аппарат. Часть 1

В этой статье речь пойдет о методе построения деревьев решений, который впервые был предложен Р. Куинленом (R. Quinlan). Этот метод используется в одном из лучших алгоритмов построения деревьев решений C4.5.

Прежде чем приступить к описанию алгоритма построения дерева решений, определим обязательные требования к структуре данных и непосредственно к самим данным, при выполнении которых алгоритм C4.5 будет работоспособен:

* **Описание атрибутов.** Данные, необходимые для работы алгоритма, должны быть представлены в виде плоской таблицы. Вся информация об объектах (далее примеры) из предметной области должна описываться в виде конечного набора признаков (далее атрибуты). Каждый атрибут должен иметь дискретное или числовое значение. Сами атрибуты не должны меняться от примера к примеру, и количество атрибутов должно быть фиксированным для всех примеров.
* **Определенные классы.** Каждый пример должен быть ассоциирован с конкретным классом, т.е. один из атрибутов должен быть выбран в качестве метки класса.
* **Дискретные классы.** Классы должны быть дискретными, т.е. иметь конечное число значений. Каждый пример должен однозначно относиться к конкретному классу. Случаи, когда примеры принадлежат к классу с вероятностными оценками, исключаются. Количество классов должно быть значительно меньше количества примеров.

Алгоритм построения дерева

Пусть нам задано множество примеров T, где каждый элемент этого множества описывается m атрибутами. Количество примеров в множестве T будем называть мощностью этого множества и будем обозначать |T|.

Пусть метка класса принимает следующие значения C1, C2 … Ck.

Наша задача будет заключаться в построении иерархической классификационной модели в виде дерева из множества примеров T. Процесс построения дерева будет происходить сверху вниз. Сначала создается корень дерева, затем потомки корня и т.д.   
На первом шаге мы имеем пустое дерево (имеется только корень) и исходное множество T (ассоциированное с корнем). Требуется разбить исходное множество на подмножества. Это можно сделать, выбрав один из атрибутов в качестве проверки. Тогда в результате разбиения получаются n(по числу значений атрибута) подмножеств и, сответственно, создаются n потомков корня, каждому из которых поставлено в соответствие свое подмножество, полученное при разбиении множества T. Затем эта процедура рекурсивно применяется ко всем подмножествам (потомкам корня) и т.д.

Рассмотрим подробнее критерий выбора атрибута, по которому должно пойти ветвление. Очевидно, что в нашем распоряжении m (по числу атрибутов) возможных вариантов, из которых мы должны выбрать самый подходящий. Некоторые алгоритмы исключают повторное использование атрибута при построении дерева, но в нашем случае мы таких ограничений накладывать не будем. Любой из атрибутов можно использовать неограниченное количество раз при построении дерева.

Пусть мы имеем проверку X (в качестве проверки может быть выбран любой атрибут), которая принимает n значений A1, A2 ... An. Тогда разбиение T по проверке X даст нам подмножества T1, T2 ... Tn, при X равном соответственно A1, A2… An. Единственная доступная нам информация – то, каким образом классы распределены в множестве T и его подмножествах, получаемых при разбиении по X. Именно этим мы и воспользуемся при определении критерия.

Пусть freq(Cj,S)freq(Cj,S) – количество примеров из некоторого множества S, относящихся к одному и тому же классу Cj. Тогда вероятность того, что случайно выбранный пример из множества S будет принадлежать к классу Cj

P=freq(Cj,S)∣S∣P=freq(Cj,S)∣S∣

Согласно теории информации, количество содержащейся в сообщении информации, зависит от ее вероятности

log2(1P)log2⁡(1P), (1)

Поскольку мы используем логарифм с двоичным основанием, то выражение (1) дает количественную оценку в битах.

Выражение

Info(T)=−∑j=1kfreq(Cj,T)∣T∣∗log2freq(Cj,T)∣T∣Info(T)=−∑j=1kfreq(Cj,T)∣T∣∗log2⁡freq(Cj,T)∣T∣, (2)

дает оценку среднего количества информации, необходимого для определения класса примера из множества T. В терминологии теории информации выражение (2) называется энтропией множества T.

Ту же оценку, но только уже после разбиения множества T по X, дает следующее выражение:

Infox(T)=∑i=1n∣Ti∣∣T∣∗Info(Ti)Infox(T)=∑i=1n∣Ti∣∣T∣∗Info(Ti), (3)

Тогда критерием для выбора атрибута будет являться следующая формула:

Gain(X)=Info(T)−Infox(T)Gain(X)=Info(T)−Infox(T), (4)

Критерий (4) считается для всех атрибутов. Выбирается атрибут, максимизирующий данное выражение. Этот атрибут будет являться проверкой в текущем узле дерева, а затем по этому атрибуту производится дальнейшее построение дерева. Т.е. в узле будет проверяться значение по этому атрибуту и дальнейшее движение по дереву будет производиться в зависимости от полученного ответа.

Такие же рассуждения можно применить к полученным   
подмножествам T1, T2 … Tn и продолжить рекурсивно процесс построения дерева, до тех пор, пока в узле не окажутся примеры из одного класса.

Одно важное замечание: если в процессе работы алгоритма получен узел, ассоциированный с пустым множеством (т.е. ни один пример не попал в данный узел), то он помечается как лист, и в качестве решения листа выбирается наиболее часто встречающийся класс у непосредственного предка данного листа.

Здесь следует пояснить почему критерий (4) должен максимизироваться. Из свойств энтропии нам известно, что максимально возможное значение энтропии достигается в том случае, когда все его сообщения равновероятны. В нашем случае, энтропия (3) достигает своего максимума когда частота появления классов в примерах множества T равновероятна. Нам же необходимо выбрать такой атрибут, чтобы при разбиении по нему один из классов имел наибольшую вероятность появления. Это возможно в том случае, когда энтропия (3) будет иметь минимальное значение и, соответственно, критерий (4) достигнет своего максимума.

Как быть в случае с числовыми атрибутами? Понятно, что следует выбрать некий порог, с которым должны сравниваться все значения атрибута. Пусть числовой атрибут имеет конечное число значений. Обозначим их {v1, v2 … vn}. Предварительно отсортируем все значения. Тогда любое значение, лежащее между viи vi+1, делит все примеры на два множества: те, которые лежат слева от этого значения {v1, v2 … vi}, и те, что справа {vi+1, vi+2 … vn}. В качестве порога можно выбрать среднее между значениями vi и vi+1

THi=vi+vi+12THi=vi+vi+12

Таким образом, мы существенно упростили задачу нахождения порога, и привели к рассмотрению всего n-1 потенциальных пороговых значений TH1, TH2 … THn-1.   
Формулы (2), (3) и (4) последовательно применяются ко всем потенциальным пороговым значениям и среди них выбирается то, которое дает максимальное значение по критерию (4). Далее это значение сравнивается со значениями критерия (4), подсчитанными для остальных атрибутов. Если выяснится, что среди всех атрибутов данный числовой атрибут имеет максимальное значение по критерию (4), то в качестве проверки выбирается именно он.

Следует отметить, что все числовые тесты являются бинарными, т.е. делят узел дерева на две ветви.

Классификация новых примеров

Итак, мы имеем дерево решений и хотим использовать его для распознавания нового объекта. Обход дерева решений начинается с корня дерева. На каждом внутреннем узле проверяется значение объекта Y по атрибуту, который соответствует проверке в данном узле, и, в зависимости от полученного ответа, находится соответствующее ветвление, и по этой дуге двигаемся к узлу, находящему на уровень ниже и т.д. Обход дерева заканчивается как только встретится узел решения, который и дает название класса объекта Y.

Итак, первый шаг сделан. Вы можете легко получить дерево решений, реализовав описанный алгоритм. Правда, небольшим недостатком будут ветвистые деревья, если примеры представлены очень большим количеством атрибутов.

Этот недостаток можно устранить, применив методику отсечения (pruning) ветвей, но это уже вопрос отдельной статьи.