Специальные вопросы построения деревьев решений

6.5.4. Большое дерево не означает, что оно "подходящее"

Чем больше частных случаев описано в дереве решений, тем меньшее количество объектов попадает в каждый частный случай. Такие деревья называют "ветвистыми" или "кустистыми", они состоят из неоправданно большого числа узлов и *ветвей*, исходное множество разбивается на большое число подмножеств, состоящих из очень малого числа объектов. В результате "переполнения" таких деревьев их способность к обобщению уменьшается, и построенные модели не могут давать верные ответы.

В процессе построения дерева, чтобы его размеры не стали чрезмерно большими, используют специальные процедуры, которые позволяют создавать оптимальные деревья, так называемые деревья "подходящих размеров".

Какой размер дерева может считаться оптимальным? Дерево должно быть достаточно сложным, чтобы учитывать информацию из исследуемого набора данных, но одновременно оно должно быть достаточно простым. Другими словами, дерево должно использовать информацию, улучшающую качество модели, и игнорировать ту информацию, которая ее не улучшает.

Тут существует две возможные стратегии. Первая состоит в наращивании дерева до определенного размера в соответствии с параметрами, заданными пользователем. Определение этих параметров может основываться на опыте и интуиции аналитика, а также на некоторых "диагностических сообщениях" системы, конструирующей дерево решений.

Вторая стратегия состоит в использовании набора процедур, определяющих "подходящий размер" дерева, они разработаны Бриманом, Куилендом и др. в 1984 году. Однако, как отмечают авторы, нельзя сказать, что эти процедуры доступны начинающему пользователю.

Процедуры, которые используют для предотвращения создания чрезмерно больших деревьев, включают: сокращение дерева путем отсечения ветвей; использование правил остановки обучения.

Следует отметить, что не все алгоритмы при конструировании дерева работают по одной схеме. Некоторые алгоритмы включают два отдельных последовательных этапа: *построение* дерева и его *сокращение*; другие чередуют эти этапы в процессе своей работы для предотвращения наращивания *внутренних узлов*.

6.5.5. Остановка построения дерева

Рассмотрим *правило* остановки. Оно должно определить, является ли рассматриваемый узел *внутренним узлом*, при этом он будет разбиваться дальше, или же он является *конечным узлом*, т.е. *узлом решения*.

Остановка – такой момент в процессе построения дерева, когда следует прекратить дальнейшие ветвления.

Один из вариантов *правил* остановки – "ранняя остановка" (prepruning), она определяет целесообразность разбиения узла. Преимущество использования такого варианта – уменьшение времени на обучение модели. Однако здесь возникает риск снижения точности классификации. Поэтому рекомендуется "вместо остановки использовать отсечение".

Второй вариант остановки обучения – ограничение глубины дерева. В этом случае построение заканчивается, если достигнута заданная глубина.

Еще один вариант остановки – задание минимального количества примеров, которые будут содержаться в *конечных узлах* дерева. При этом варианте ветвление продолжаются до того момента, пока все *конечные узлы* дерева не будут чистыми или будут содержать не более чем заданное число объектов.

Существует еще ряд *правил*, но ни одно из них не имеет большой практической ценности, а некоторые применимы лишь в отдельных случаях.

6.5.6. Сокращение дерева или отсечение ветвей

Решением проблемы слишком ветвистого дерева является его *сокращение* путем отсечения (*pruning*) некоторых *ветвей*.

Качество классификационной модели, построенной при помощи дерева решений, характеризуется двумя основными признаками: точностью распознавания и ошибкой.

Точность распознавания рассчитывается как отношение объектов, правильно классифицированных в процессе обучения, к общему количеству объектов набора данных, которые принимали участие в обучении.

Ошибка рассчитывается как отношение объектов, неправильно классифицированных в процессе обучения, к общему количеству объектов набора данных, которые принимали участие в обучении.

Отсечение *ветвей* или замену некоторых *ветвей* под деревом следует проводить там, где эта процедура не приводит к возрастанию ошибки. Процесс проходит снизу вверх, т.е. является восходящим. Это более популярная процедура, чем использование *правил* остановки. Деревья, получаемые после отсечения некоторых *ветвей*, называют усеченными.

Если такое усеченное дерево все еще не является интуитивным и сложным для понимания, используют извлечение *правил*, которые объединяют в наборы для описания классов. Каждый путь от корня дерева до его вершины или листа дает одно *правило*. Условиями *правила* являются проверки на *внутренних узлах* дерева.

6.6. Алгоритмы деревьев решений

На сегодняшний день существует большое число алгоритмов, реализующих деревья решений: *CART*, C4.5, CHAID, CN2, NewId, ITrule и другие.

Алгоритм CART

Алгоритм CART (Classification and Regression Tree), как видно из названия, решает задачи классификации и регрессии. Он разработан в 1974-1984 годах четырьмя профессорами статистики – Leo Breiman (Berkeley), JerryFriedman (Stanford), CharlesStone (Berkeley) и RichardOlshen (Stanford).

Атрибуты набора данных могут иметь как категориальное, так и числовое значение.

Алгоритм CART предназначен для построения бинарного дерева решений. Бинарные деревья также называют двоичными. Пример такого дерева рассматривался в начале лекции.

Другие особенности алгоритма *CART*:

- функция оценки качества разбиения;
- механизм отсечения дерева;
- алгоритм обработки пропущенных значений;
- построение деревьев регрессии.

Каждый узел бинарного дерева при разбиении имеет только двух потомков, называемых дочерними ветвями. Дальнейшее разделение ветви зависит от того, много ли исходных данных описывает данная *ветвь*. На каждом шаге построения дерева *правило*, формируемое в узле, делит заданное множество примеров на две части. Правая его часть (*ветвь* right) – это та часть множества, в которой *правило* выполняется; левая (*ветвь* left) – та, для которой *правило* не выполняется.

Функция оценки качества разбиения, которая используется для выбора оптимального *правила,* – индекс Gini – был описан выше. Отметим, что данная оценочная функция основана на идее уменьшения неопределенности в узле. Допустим, есть узел, и он разбит на два класса. Максимальная неопределенность в узле будет достигнута при разбиении его на два подмножества по 50 примеров, а максимальная определенность – при разбиении на 100 и 0 примеров.

Правила разбиения. Напомним, что алгоритм *CART* работает с числовыми и категориальными атрибутами. В каждом узле разбиение может идти только по одному атрибуту. Если атрибут является числовым, то во *внутреннем узле* формируется *правило* вида $x_i <= c$, Значение c в большинстве случаев выбирается как среднее арифметическое двух соседних упорядоченных значений переменной x_i обучающего набора данных. Если же атрибут относится к категориальному типу, то во *внутреннем узле* формируется *правило* x_i $V(x_i)$, где $V(x_i)$ – некоторое непустое подмножество множества значений переменной x_i в обучающем наборе данных.

Механизм отсечения. Этим механизмом, имеющим название *minimal* cost-complexity tree *pruning*, алгоритм *CART* принципиально отличается от других алгоритмов конструирования деревьев решений. В рассматриваемом алгоритме отсечение – это некий компромисс между получением дерева "подходящего размера" и получением наиболее точной оценки классификации. Метод

заключается в получении последовательности уменьшающихся деревьев, но деревья рассматриваются не все, а только "лучшие представители".

Перекрестная проверка (V-*fold* cross-validation) является наиболее сложной и одновременно оригинальной частью алгоритма *CART*. Она представляет собой путь выбора окончательного дерева, при условии, что набор данных имеет небольшой объем или же записи набора данных настолько специфические, что разделить набор на обучающую и тестовую выборку не представляется возможным.

Итак, основные характеристики алгоритма *CART*: бинарное расщепление, *критерий расщепления* – индекс Gini, алгоритмы *minimal* cost-complexity tree *pruning* и V-*fold* cross-validation, принцип "вырастить дерево, а затем сократить", высокая скорость построения, обработка пропущенных значений.

Алгоритм C4.5

Алгоритм C4.5 строит дерево решений с неограниченным количеством *ветвей* у узла. Данный алгоритм может работать только с дискретным зависимым атрибутом и поэтому может решать только задачи классификации. C4.5 считается одним из самых известных и широко используемых алгоритмов построения деревьев классификации.

Для работы алгоритма C4.5 необходимо соблюдение следующих требований:

- Каждая запись набора данных должна быть ассоциирована с одним из предопределенных классов, т.е. один из атрибутов набора данных должен являться меткой класса.
- Классы должны быть дискретными. Каждый пример должен однозначно относиться к одному из классов.
- Количество классов должно быть значительно меньше количества записей в исследуемом наборе данных.

Алгоритм C4.5 медленно работает на сверхбольших и зашумленных наборах данных.

Мы рассмотрели два известных алгоритма построения деревьев решений CART и C4.5. Оба алгоритма являются робастными, т.е. устойчивыми к шумам и выбросам данных.

Таким образом, алгоритмы построения деревьев решений различаются следующими характеристиками:

- вид расщепления – бинарное (binary), множественное (multi-way),
- *критерии расщепления* – энтропия, Gini, другие,
- возможность обработки пропущенных значений,
- процедура сокращения *ветвей* или отсечения,
- возможности извлечения *правил* из деревьев.

Ни один алгоритм построения дерева нельзя априори считать наилучшим или совершенным, подтверждение целесообразности использования конкретного алгоритма должно быть проверено и подтверждено экспериментом.

Выводы

В лекции мы рассмотрели метод деревьев решений. Определить его кратко можно как иерархическое, гибкое средство предсказания принадлежности объектов к определенному классу или прогнозирования значений числовых переменных.

Качество работы рассмотренного метода деревьев решений зависит как от выбора алгоритма, так и от набора исследуемых данных. Несмотря на все преимущества данного метода, следует помнить, что для того, чтобы построить качественную модель, необходимо понимать природу взаимосвязи между зависимыми и независимыми переменными и подготовить достаточный набор данных.