Глава 26

ДЕРЕВЬЯ КЛАССИФИКАЦИИ И РЕГРЕССИИ

В ЭТОЙ ГЛАВЕ...

- Описание структуры деревьев решений
- Классификация и регрессия
- Формирование дерева
- Процедура обучения
- Обсуждение полученных результатов
- Резюме

Деревья регрессии и классификации, известные также под общим названием как деревья решений (Decision Tree — DT), представляют собой структуры данных, позволяющие интерпретировать шаблоны данных с целью их распознавания. Деревья решений организованы в виде иерархической структуры, состоящей из узлов принятия решений по оценке значений определенных переменных для прогнозирования результирующего значения. Применение деревьев классификации приводит к получению символического обозначения класса, а в результате использования деревьев регрессии происходит возврат непрерывных значений.

Для обучения деревьев решений должны быть предусмотрены примеры данных, поэтому необходимо создавать или собирать такие данные заранее. С одной стороны, данные могут быть подготовлены экспертом, а с другой стороны, может быть предусмотрено накопление коллекции фактов, касающихся рассматриваемой задачи. На основе исключительно только интерпретации (или классификации) подобных данных могут быть созданы многие приложения, в том числе модели ИИ. На практике каждая рассматриваемая при этом задача может быть представлена с помощью множества атрибутов, применительно к которому дерево решений прогнозирует неизвестный атрибут (решение).

В настоящей главе обсуждаются перечисленные ниже темы.

- Определение понятия выборки данных, которая включает переменные прогнозирования и отклика.

- Описание структуры деревьев решений, состоящих из узлов принятия решений и ребер.
- Организация процесса классификации и регрессии на основе существующего дерева.
- Определение способов обучения (или формирования на основе логического вывода) деревьев решений на базе наборов данных, с использованием рекурсивного секционирования.
- Описание применения процедуры обучения для решения более общей задачи, цель которой состоит в поиске дерева решений, которое в наибольшей степени соответствует рассматриваемой задаче.

Деревья решений могут использоваться в качестве компонента, отвечающего за принятие решений в ходе реализации персонажей игры. Каждая ситуация представляется в виде множества атрибутов, поэтому в каждой конкретной ситуации дерево решений может предложить наилучший способ действий. Кроме того, деревья регрессии могут использоваться для оценки сильных сторон любого объекта (или для прогнозирования результата, как положительного, так и отрицательного). В следующей главе эта возможность используется в контексте выбора оружия.

Описание структуры деревьев решений

Прежде чем приступать к рассмотрению того, что позволяют достичь деревья решений, или даже к рассмотрению вопроса об их устройстве, необходимо понять основные концепции, лежащие в основе такой структуры данных, как дерево решений.

Любое дерево решений выводит прогнозируемое значение, полученное в результате оценки некоторых входных атрибутов. Деревья решений подразделяются на два разных типа: деревья классификации и деревья регрессии. Это различие не зависит от типов входных данных, поскольку деревья того и другого типов могут принимать либо непрерывные, либо символические значения. Определяющим фактором, от которого зависит тип дерева, является выходное значение. Дерево решений с непрерывными выходными значениями именуется деревом регрессии, а деревья классификации вместо этого выводят конкретные значения. Итоговые сведения, определяющие различия между типами деревьев решений, приведены в табл. 26.1.

Таблица 26.1. Различие между двумя типами деревьев решений, определяемое тем, результат какого типа они возвращают

Тип дерева	Прогнозирование	Тип данных
Дерево классификации	Дискретное	Символы
Дерево регрессии	Непрерывное	Вещественные числа

В первом подразделе данной главы приведено определение понятия выборки данных и показана его значимость. Выборки данных предназначены для обработки деревьями решений и часто служат также в качестве определения формата ввода-вывода. В следующих разделах рассматриваются способы представления структуры

деревьев решений и раскрываются секреты того, благодаря чему деревья решений становятся столь простыми и быстродействующими. Наконец, более подробно исследуются наиболее важные части деревьев решений, такие как узлы принятия решений и листовые узлы.

Общие сведения о выборках данных

Безусловно, без применения данных становится невозможным не только распознавание образов, но и в целом машинное обучение. Совокупности данных часто представляют в виде отдельных выборок. Такие выборки иногда называют *событиями*, *экземплярами*, *шаблонами*, а также, безусловно, применяют для их обозначения многие другие названия.

По существу, любая выборка данных представляет собой множество атрибутов, которые принято также называть переменными прогнозирования. Каждый атрибут может представлять собой непрерывное значение (т.е. число с плавающей точкой) или символ (т.е. множество неупорядоченных дискретных значений). Такие атрибуты позволяют концептуально представить почти любую информацию; в контексте выбора оружия в качестве атрибутов рассматриваются такие свойства, как вес, скорострельность и максимальное количество боеприпасов.

Предусмотрена также возможность применять дополнительные атрибуты, имеющие специальное значение, которые известны под названием переменных отклика (или зависимых переменных). Такой атрибут может быть выражен с помощью символа, представляющего дискретные категории (в деревьях классификации) или непрерывное значение (в деревьях регрессии). Переменные отклика могут служить в качестве критериев для принятия решений по отношению к каждой из выборок при решении задач обоих типов — и классификации, и регрессии.

В табл. 26.2 приведен пример данных, относящихся к задаче прогнозирования общего причиненного ущерба исходя из свойств оружия. В качестве переменной отклика используется ущерб, поэтому остальные переменные выполняют роль переменных прогнозирования. Значения веса и типа заданы как конкретные данные, а остальные значения являются непрерывными.

Таблица 26.2. Четыре выборки данных с многочисленными атрибутами

Вес	Скорострельность	Емкость	Дальность стрельбы	Тип	Ущерб
Легкий	47	10	40 м	Пистолет	5%
Тяжелый	200	500	100 м	Автомат	10%
Очень легкий	6	6	25 м	Пистолет	4%
Очень тяжелый	280	1000	200 м	Автомат	13%

Во всем остальном между переменными прогнозирования и отклика нет никаких различий; в качестве переменной отклика может использоваться почти любой атрибут. Переменная, используемая в качестве основы для процесса классификации, может быть выбрана с учетом специфики каждой задачи. Например, для классификации типов оружия можно, кроме всего прочего, выбрать атрибут, определяющий ущерб или мощность.

ПРИМЕЧАНИЕ

Выборки данных обычно используются во всех областях ИИ, включая все прочие методы, которые могут служить для распознавания образов и прогнозирования. Особенно ярким примером этого являются нейронные сети. Пояснения по поводу проблемы классификации были приведены в главе 17, "Персептроны", и в главе 19, "Многослойные персептроны" (в этих главах основное внимание было уделено аппроксимации функций), но основная часть концепций, рассматриваемых в настоящей главе, может непосредственно применяться к нейронным сетям.

Деревья решений

Любое дерево решений по существу представляет собой древовидный граф, в буквальном смысле этого понятия, сформулированного в компьютерных науках. Эта структура данных состоит из узлов, соединенных друг с другом ребрами (рис. 26.1). При этом не допускается, чтобы ребра образовывали цикл, так как в противном случае дерево превращается в граф, отличный от древовидного (а при использовании такого графа для принятия решений возникают затруднения).

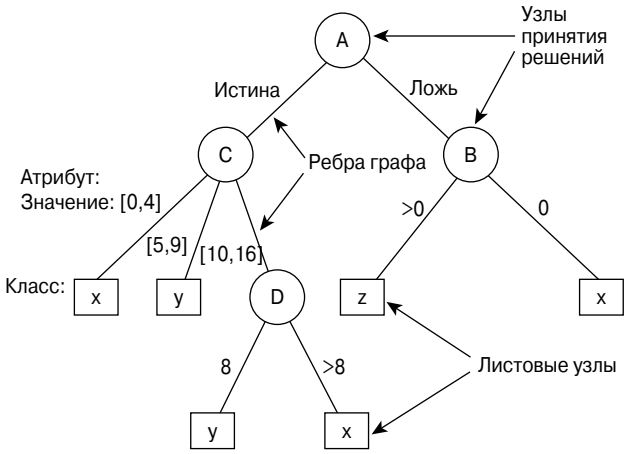

Рис. 26.1. *Простое дерево решений, в котором корневой узел показан сверху, а каждый узел принятия решений обозначен кружком; листовые узлы обозначаются прямоугольниками*

В дереве имеется один особый узел, известный как *корневой*. По существу, этот узел является основой дерева, поскольку от корня можно перейти по дереву к любому другому узлу. Еще к одной особой разновидности узлов относятся узлы, находящиеся в конце любой цепочки подряд идущих ребер, — *листовые узлы*.

Приведенное выше описание древовидной структуры данных является очень широким. С этими сведениями должен быть знаком любой специалист, имеющий приемлемый опыт работы с языками программирования. Но если речь идет об использовании деревьев решений в искусственном интеллекте, то каждое из изложенных выше понятий приобретает особый смысл.

Каждый уровень в дереве может рассматриваться как одно из решений; узел принятия решений обеспечивает проверку условия, а каждое ребро обозначает один из возможных вариантов. Более формально можно отметить, что узлы принятия реше-

ний содержат критерии выбора, а ребра выражают взаимоисключающие результаты проверки соответствия этим критериям.

По существу, при каждой проверке условия происходит сортировка выборок данных таким образом, что каждый элемент данных определяется как соответствующий только одному ребру. Если все выборки рассматриваются как одно общее множество данных, то критерии принятия решений разбивают это множество на непересекающиеся подмножества, как показано на рис. 26.2. В результате объединения таких проверок в некоторую иерархию фактически организуется процесс разбиения всех данных на все меньшие части, происходящий до тех пор, пока не достигается листовой узел. Каждый листовой узел соответствует небольшой, но исключительной (неповторяющейся) части исходного множества.

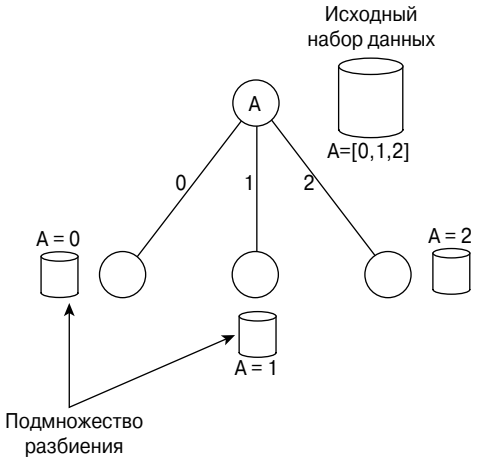

Рис. 26.2. *Разбиение выборки данных на взаимоисключающие подмножества с использованием узла принятия решений*

Способы проверки условий

Нет никакого сомнения в том, что количество возможных способов представления решений должно быть весьма велико, поэтому выбор применяемого способа зависит от типа проверяемого атрибута, а также от операции, используемой в проверке условия. Поскольку атрибуты могут выражаться в виде символов или непрерывных значений, сами проверки могут быть организованы в виде булевых условий или непрерывных отношений. А от количества возможных результатов проверки зависит то, сколько ребер должно исходить из рассматриваемого узла принятия решений. Ниже перечислены наиболее часто применяемые проверки условий.

- **Проверка булева значения.** При проведении такой проверки определяется то, приводит ли применение какого-то конкретного оператора к получению истинного или ложного значения. Очевидно, что возможными результатами проверки становятся истина или ложь.
- **Определение знака.** При выполнении такой проверки определяется знак выражения. Результатом может быть либо положительное, либо отрицательное

значение. Указанная проверка может рассматриваться как частный случай проверки булева значения.

- **Проверка принадлежности к классу.** При выполнении такой проверки определяется, к какому классу принадлежит данный символ. Результатом проверки становится обозначение одного из возможных классов (количество которых может быть произвольным).
- **Проверка принадлежности к области значений.** При проведении такой проверки должно быть выяснено, к какой области значений относится данное значение. Каждый из возможных результатов указывает, к какой области значений, на которые делится вся область значений переменной, относится данное значение. Такая проверка может рассматриваться как проверка принадлежности к классу.

Как правило, в каждом узле принятия решений выполняется единственная проверка условия, охватывающая только один атрибут (в качестве примера можно указать операцию проверки `B==true`). Такой подход часто становится вполне приемлемым, поскольку дерево решений позволяет создавать иерархические комбинации проверок для формирования более сложных структур принятия решений. Однако некоторые более сложные разновидности деревьев решений допускают проведение проверок условий, в которых рассматривается большее количество атрибутов (например, одновременно проверяются условия `A==1` и `B<0`), что позволяет повысить точность проверок, за счет усложнения. Но в этом случае возрастает количество возможных комбинаций, поэтому увеличивается количество исходящих ребер, как показано в табл. 26.3.

Таблица 26.3. Четыре примера простых проверок условий и операций выработки решений с использованием большего количества атрибутов

Проверка условия	Возможные результаты
`A` (булево значение)	`true`
	`false`
`B` (знак)	`B > 0`
	`B ≤ 0`
`C` (принадлежность к диапазону)	`C in [0..4]`
	`C in [5..9]`
	`C in [10..14]`
`A, B`	`A == true and B > 0`
	`A == false and B > 0`
	`A == true and B ≤ 0`
	`A == false and B ≤ 0`

Вообще говоря, увеличение количества переменных прогнозирования, используемых при проверке условия в узле принятия решений, приводит к увеличению количества исходящих ребер. Но не исключена возможность применять произвольное

количество атрибутов и отображать их на единственное булево решение. Для этого может потребоваться вычислять в узле принятия решений такие выражения, как `(A and B) or C`. Аналогичные выражения могут также использоваться для преобразования непрерывных значений в единственное число. Но недостатком подхода, предусматривающего использование подобных сложных выражений, является то, что при этом задача обучения дерева решений становится сложнее!

Листовые узлы и переменные отклика

Кроме того, в деревьях решений предусмотрено применение таких отдельных переменных отклика, которые относятся только к листовым узлам дерева. Эти значения переменных отклика также могут быть либо дискретными, либо непрерывными.

Каждый из листовых узлов соответствует одному конкретному подмножеству значений выборок данных (рис. 26.3). В деревьях классификации переменная отклика соответствует прогнозируемой категории для всех выборок, отображаемых на этот листовой узел. С другой стороны, непрерывные переменные отклика, применяемые в деревьях регрессии, обычно показывают среднее значение для всех выборок, соответствующих данному листовому узлу.

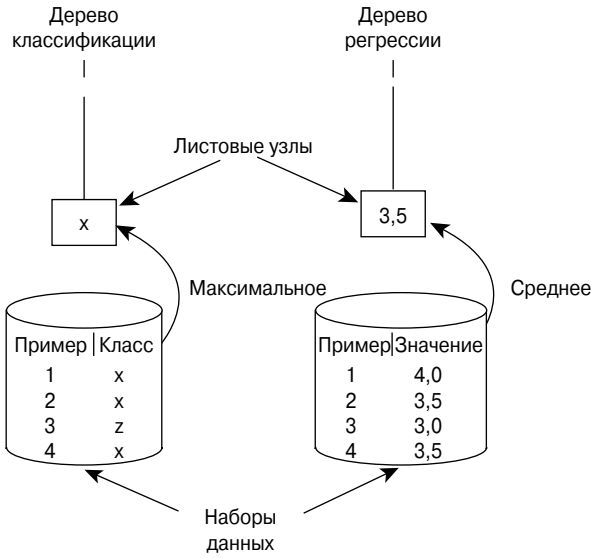

Рис. 26.3. *Соответствие между значениями, хранящимися в каждом листовом узле, и переменными отклика, для тех выборок, которые отображаются на этот узел*

Очевидно, что структура дерева решений не зависит от выбранного типа переменных отклика, так как от типа переменных отклика, прежде всего, зависит то, какой способ хранения данных (условно говоря, контейнер для данных) будет использоваться в листовых узлах. От типа переменных отклика существенным образом зависит также возможный способ формирования дерева, но эта тема будет рассматриваться после того, как мы ознакомимся со способами использования деревьев решений.

Классификация и регрессия

После создания дерева решений появляется возможность оценивать значение переменной отклика для неизвестных выборок данных с учетом лишь значений атрибутов. При выполнении такой операции одна выборка обрабатывается за другой, что позволяет проверять на соответствие критериям и оценивать целые наборы данных.

Обработка каждой выборки данных и, в частности, определение класса и (или) значения для выборки осуществляются путем перехода по дереву, начиная от корневого узла, в направлении к листовым узлам. Каждый этап перехода осуществляется с учетом результатов проверки условий, а выбор направления перехода происходит с учетом переменных прогнозирования (рис. 26.4).

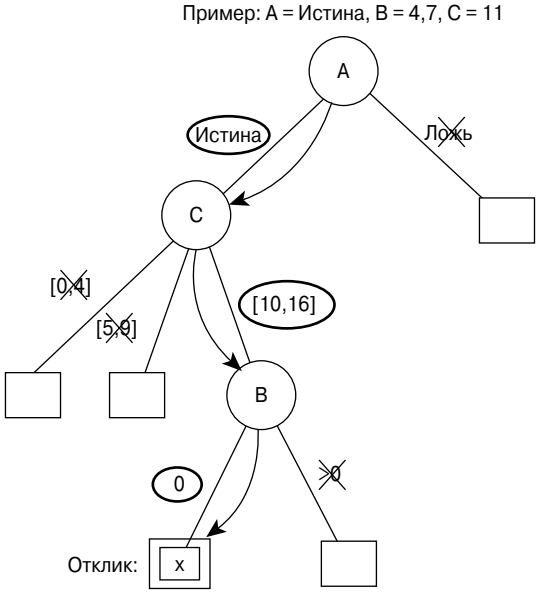

Рис. 26.4. *Использование выборки данных для управления переходом по дереву решений*

На каждом уровне дерева решение принимается на основе значений атрибутов. Для оценки значений переменных прогнозирования используются проверки условий в каждом узле. Результат оценки всегда соответствует только одному из ребер, исходящих из узла принятия решений. Наличие в любых обстоятельствах только одного применимого ребра гарантированно, поскольку условия являются полными (т.е. допускающими не меньше одного варианта решения) и взаимоисключающими (т.е. допускающими только один вариант решения). Во время перехода прослеживается ребро, ведущее к следующему узлу принятия решений, в котором этот процесс повторяется.

Переход по дереву прекращается после достижения конца цепочки ребер (листинг 26.1). Каждый листовой узел в дереве соответствует одному из классов (или определенному значению), поэтому выборка данных, которая использовалась для прохождения по дереву, должна принадлежать данному классу (или иметь данное

значение). Дерево решений гарантирует, что каждая выборка будет соответствовать одному и только одному листовому узлу, и поэтому ей будет присвоено единственное обозначение класса или одна оценка.

Листинг 26.1. Алгоритм, применяемый для перехода по дереву решений с использованием значения выборки данных

```
node = root
repeat
     result = node.evaluate( sample )
     for each branch from node
          if branch.match( result )
               node = branch.child
          end if
     end for
until node is a leaf
return leaf class or value
```

С концептуальной точки зрения, процесс применения дерева решений является чрезвычайно простым, и это позволяет понять, с чем связана его эффективность. Наибольшая сложность заключается в проектировании программного обеспечения, позволяющего реализовать какой-то гибкий способ моделирования деревьев решений.

Формирование дерева

Большинство деревьев решений имеют чрезвычайно простую структуру, к тому же, на практике обычно не достигают больших размеров, особенно в компьютерных играх. Поэтому задача формирования таких деревьев вручную вполне осуществима. Тем не менее подобный подход аналогичен созданию экспертной системы в виде иерархической структуры, но в нем игнорируются основные преимущества деревьев решений — возможность проводить их обучение автоматически.

Для решения задачи обучения деревьев решений применялось много разных схем обучения. Как правило, в этих методах используются алгоритмы формирования деревьев решений на основе предположений, принятых в результате исследований выборок данных; такой подход принято называть формированием деревьев решений. Каждый из таких методов имеет свои преимущества и недостатки. Но один из подходов стал основой большинства других решений, поскольку оказался очень быстродействующим и простым в реализации. Применяемый в этом подходе алгоритм известен под названием *рекурсивного секционирования*. По существу, этот алгоритм сводится к тому, что дерево создается инкрементно в результате пакетной обработки множества данных. Для создания узлов принятия решений и соответствующих проверок условий используются статистические данные и предпринимается попытка сформировать эффективные и компактные деревья.

Краткое описание алгоритма рекурсивного секционирования

Алгоритм, впервые предложенный Куинланом (Quinlan), действует по принципу рекурсивного секционирования набора данных и инкрементного построения дерева [64]. По такому же принципу действуют многие другие алгоритмы, поэтому вначале рассмотрим общее описание, а затем перейдем к анализу деталей этого алгоритма.

В основе алгоритма рекурсивного секционирования лежит принцип "разделяй и властвуй". Алгоритм применяется с целью формирования дерева, позволяющего распределять по категориям выборки данных; набор данных разбивается на грубо классифицированные подмножества, после чего предпринимается еще одна попытка, до тех пор пока классификация не станет идеальной (или, по крайней мере, достаточно качественной). Этот процесс является рекурсивным, и в ходе его осуществления формируется дерево решений.

Работа алгоритма начинается с пустого дерева и полного набора данных. Вначале необходимо найти приемлемую точку разбиения, чтобы разделить с помощью этой точки первоначальное множество на подмножества. Как правило, для выбора атрибута (или нескольких атрибутов), который служит для секционирования данных, применяется эвристический метод. С точки зрения реализации алгоритма, применяемый способ выбора атрибута не имеет значения, при условии что он позволяет успешно создавать узлы принятия решений и секционировать набор данных. Таким образом может быть создан первый узел в дереве, соответствующий данной проверке условия.

После первой итерации алгоритма появляется дерево с одним узлом, разбивающим набор данных на два (или больше) подмножества. После этого данный процесс может быть выполнен повторно, применительно к каждому из подмножеств, для создания поддеревьев. Процесс разбиения прекращается после того, как отпадает необходимость дальнейшей разбивки набора данных. Решение по прекращению разбивки может быть выработано с помощью той же процедуры, которая применяется для создания узлов принятия решений, как показано в листинге 26.2.

Листинг 26.2. Набросок алгоритма рекурсивного секционирования, с помощью которого предпринимается попытка создавать новые узлы принятия решений, а затем осуществлять разбивку набора данных соответствующим образом

```
function partition( dataset, node )
     if not create_decision( dataset, node )
               return
     end if
     for each sample in dataset
               result = node.evaluate( sample )
               subset[result].add( sample )
     end for
     for each result in subset
               partition( subset, child )
               node.add( branch, result )
               branch.add( child )
     end for
end function
```

Такой процесс секционирования по самой своей сути является рекурсивным, но редко позволяет достичь значений глубины, выходящих за пределы первого десятка. Но в действительности количество этапов рекурсивного выполнения алгоритма, в основном, определяется тем, насколько сложна рассматриваемая задача. В случае применения простых наборов данных работа функции `partition` прекращается быстро. Еще одним фактором, который служит критерием прекращения рекурсии, является количество атрибутов; каждый уровень в дереве представляет собой одно из решений, поэтому чем больше атрибутов могут использоваться для разбиения данных, тем больше возможное количество рекурсивных повторов алгоритма. Если в процессе применения рекурсии возникают проблемы, то рекурсивный метод может быть преоб-

разован в итерационный, основанный на применении стека, но создается впечатление, что современные компьютеры с наибольшей вероятностью позволят обработать большинство наборов данных без каких-либо затруднений (поскольку сам этот алгоритм был разработан с учетом возможностей масштабирования).

Одна и та же проверка условия никогда не используется в дереве дважды; дело в том, что дублирующаяся проверка не приведет к дальнейшему разбиению набора данных (связанный с ней узел окажется избыточным). Причем избежать проблемы появления дублирующихся проверок неявно позволяют все алгоритмы, поскольку все другие варианты проверок, по сравнению с дублирующимися, всегда оцениваются как лучшие! Но это не исключает возможности снова применить какой-то конкретный атрибут для разбиения набора данных. Необходимость в этом может возникнуть, если атрибут является непрерывным или имеет много возможных символических значений. Обработка некоторых наборов данных может осуществляться более успешно, если разбиение по одной и той же размерности осуществляется неоднократно, но в разных местах.

Разбиение наборов данных

Единственный аспект применения рассматриваемого алгоритма, который требует более глубоких размышлений, состоит в создании узлов принятия решений. Для этого необходимо при наличии определенного набора данных установить, какими должны быть критерии секционирования этого набора.

Как правило, при выборе атрибутов применяется *жадный* алгоритм, т.е. алгоритм, предусматривающий выбор наилучшего из приемлемых атрибутов. Для выявления атрибута, обеспечивающего наилучшее разбиение, может применяться статистический анализ. Разбиение множества в соответствии с таким атрибутом приводит к созданию подмножеств, характеризующихся минимальной статистической ошибкой. Тем не менее такой подход не гарантирует, что все сформированное дерево окажется оптимальным. В действительности, любой жадный алгоритм не позволяет по определению заглядывать далеко вперед, поэтому, как правило, не позволяет выработать оптимальное решение. Но несмотря на это, деревья, сформированные с помощью жадного алгоритма, позволяют добиваться приемлемых результатов (рис. 26.5).

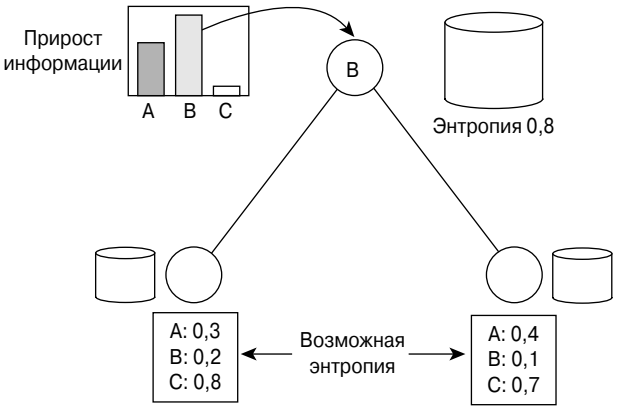

Рис. 26.5. *Рекурсивное секционирование по одному узлу, приводящее к разбиению набора данных на взаимоисключающие подмножества*

На практике пакетная обработка позволяет выявить наилучший способ разбиения. Просматриваются все выборки и измеряется количество посторонних включений в наборе применительно к каждому атрибуту. Если набор данных определяется как "содержащий посторонние включения", это означает, что набор содержит весьма разнообразные переменные отклика. Необходимо выявить все посторонние атрибуты, чтобы преобразовать их в относящиеся к набору данных, в частности, для того чтобы уменьшить разнообразие переменных отклика. Это позволяет добиться лучших результатов классификации и регрессии. Для уменьшения количества посторонних включений может применяться жадный алгоритм, позволяющий секционировать набор данных с учетом атрибута, который является для этого набора данных в наибольшей степени посторонним.

Для измерения количества посторонних включений необходимо определить расчетным путем энтропию. Понятие энтропии означает почти то же самое, что и количество посторонних включений, и если вы хотите произвести впечатление на своих руководителей или сбить их с толку, то можете вместо термина "количество посторонних включений" использовать термин "энтропия"! Ниже приведено общепринятое определение количества посторонних включений для булевых и (или) двоичных значений, взятых из множества S.

$$Entropy(s) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$$

В этой формуле p_+ обозначает относительное количество истинных значений, а p_- показывает относительное количество ложных значений. Энтропия становится равной 0, если все примеры относятся к одной и той же категории, и 1, если величины представлены в соотношении 50 %/50%. Более общая формула для множеств с количеством классов, равным c, является таковой:

$$Entropy(s) = \sum_{i=1}^{c} -p_i \log_2(p_i), \quad \text{где } p_i = \frac{|S_i|}{|S|}$$

Энтропия определяется как сумма взвешенных логарифмических значений для каждого класса i, а p_i обозначает долю выборок, относящихся к классу i (иными словами, количество выборок $|S_i|$ в общем количестве выборок в наборе данных $|S|$).

Чтобы определить, какой наилучший атрибут следует выбрать для осуществления секционирования, недостаточно просто вычислить энтропию. Если какой-то атрибут намечен для использования при разбиении, то необходимо определить прирост информации, измеряющий ожидаемое уменьшение энтропии:

$$Gain(S,A) = Entropy(S) - \sum_{i=1}^{c} p_i \, Entropy(S_i)$$

Эта формула выражает прирост информации в виде суммарной энтропии множества, но за вычетом энтропии подмножеств, созданных в результате разбиения. После приведения такого метода к процедурной форме может быть создан алгоритм, псевдокод которого показан в листинге 26.3.

Листинг 26.3. Функция, используемая для поиска наилучшего атрибута, обеспечивающего секционирование набора данных, если таковой атрибут имеется

```
function create_decision( dataset, node )
     max = 0
     # Определить количество посторонних включений для всех обучающих
данных
     entropy = compute_entropy( dataset )
     for each attribute in dataset
          # Включить разбиение и вычислить общую энтропию подмножеств
          e = entropy - compute_entropy_split( attribute, dataset )
          # Найти наилучшие положительное приращение
          if e > max
               max = e
               best = attribute
          end if
     end for
     # Создать проверку при наличии качественного атрибута
     if best
               node.evaluation = create_test( attribute )
     # В противном случае найти значение листового узла
     else
               node.class = find_class( dataset )
     end if
end function
```

Процедура обучения

При создании дерева решений окончательный результат во многом зависит от самого применяемого набора данных. Но это, вообще говоря, нежелательно, поскольку нам необходимо, чтобы результат отражал суть рассматриваемой задачи, а не характеризовал используемые данные.

Управление набором данных

Подход, предусматривающий управление набором данных, особенно важен при создании деревьев решений, когда требуется предотвратить в ходе применения алгоритма обучения возможность создания слишком конкретизированных моделей. Как и при создании многослойных персептронов, появление чрезмерно конкретизированной модели характеризуется симптомом, известным под названием *чрезмерно тщательной подгонки*. Дерево решений, отличающееся такой особенностью, становится недостаточно общим, чтобы обеспечить обработку других данных.

Для манипулирования наборами данных в методы создания деревьев решений могут быть внесены многочисленные усовершенствования, которые фактически сводятся к несложным приемам. А фундаментальные понятия остаются такими же, как и при обучении персептронов. Прежде всего необходимо подготовить три различных набора данных, которые перечислены ниже.

- Обучающее множество, которое используется в алгоритме рекурсивного секционирования.

- Аттестационное множество, которое позволяет осуществить процесс усовершенствования дерева решений, полученного в результате обучения (особенно важным усовершенствованием является отсечение лишних частей дерева).
- Проверочное множество, позволяющее выполнить окончательную проверку результатов обучения для подтверждения применимости полученного дерева решений.

Разделение всего набора данных может быть выполнено на равные части, в которых примерно треть выборок введено случайным образом. Размеры полученных множеств, в основном, зависят от общего объема имеющихся данных, но количество данных, требуемых для обучения, не может быть меньше определенного минимума. Аттестация проводится намного успешнее при значительно большем количестве выборок, а проверка при желании может быть пропущена, если наблюдается нехватка данных!

Отсечение ребер

Как правило, отсечение частей дерева решений осуществляется на этапе обработки, который следует за обучением. Для отсечения очень важно использовать отдельный набор данных, поскольку применение того же набора данных, который рассматривался во время обучения, не будет указывать на необходимость каких-либо изменений! Для отсечения используются аттестационные множества. На уровне интуитивного понимания процедуру отсечения можно описать так, что если вслед за отсечением какого-то ребра результаты классификации становятся лучше, то данное ребро необходимо отсечь.

Чтобы выполнить такое сравнение результатов, необходимо знать, какое значение переменная отклика имела бы в каждом узле принятия решений, если бы этот узел был листовым. Определение расчетным путем соответствующего класса (или значения) может осуществляться таким же образом, как и для листового узла; используется класс с наибольшим количеством экземпляров (или средним значением).

После этого выполняется проверка применительно ко всему набору данных. В каждом узле подсчитывается количество случаев успешной классификации (или общее количество ошибок регрессии). После завершения такой обработки может быть выполнена простая проверка того, имеет ли какой-либо узел лучшую оценку, чем все его дочерние узлы, вместе взятые. Если дело обстоит таким образом, то дочерние узлы фактически не нужны, поэтому в рассматриваемом узле может быть выполнено усечение дерева (рис. 26.6).

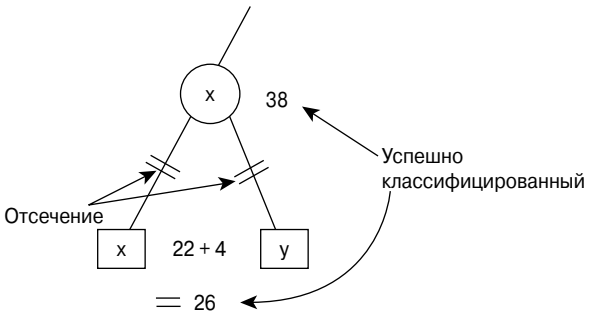

Рис. 26.6. *Отсечение ребра после подсчета количества успешно классифицированных примеров в каждом узле*

Конгломерация и усиление

В основе методов конгломерации и усиления лежит такая идея, что классификаторы низкого качества могут применяться в сочетании для создания более качественных классификаторов. Дело в том, что могут создаваться целые коллекции деревьев решений с помощью многочисленных различных параметров и обучающих множеств. Например, на этапе конгломерации может произвольно осуществляться извлечение из обучающего множества различных выборок, а на этапе усиления — присваивание им весовых коэффициентов, влияющих на обучение. В таком случае, как описано ниже, для комбинирования решений могут использоваться два различных подхода.

- Метод конгломерации действует по принципу подсчета голосов. Комбинированный классификатор выводит данные о классе, за который подано большинство голосов отдельными деревьями классификации.
- В отличие от этого, метод усиления действует по принципу присваивания каждому из деревьев весовых коэффициентов исходя из того, насколько успешным оказалось применение рассматриваемого дерева при классификации аттестационного множества. В качестве выходного значения комбинированного классификатора используется сумма взвешенных голосов отдельных деревьев классификации.

Преимуществом применения этих методов является то, что качество окончательного решения, полученного с помощью классификатора, обычно становится выше. Но эти результаты получены исключительно эмпирическим путем. Нет никакой гарантии того, что комбинированные классификаторы проявят себя как самые лучшие во время решения всей задачи в целом, хотя для закрытых множеств предложено доказательство, что полученные с их помощью результаты должны быть не хуже всех прочих.

Затраты на создание комбинированных классификаторов возрастают линейно с увеличением количества используемых деревьев решений. Для хранения структур данных требуются дополнительные объемы памяти, а выработка каждого решения связана с дополнительными затратами на прохождение по каждому из деревьев.

Обсуждение полученных результатов

Деревья решений проявляют себя как основа чрезвычайно эффективного метода вычисления оценок для неизвестных выборок. Лежащая в их основе структура данных является простой и небольшой, а для прохождения по дереву решений требуется мало усилий. Деревья решений особенно хорошо подходят для задач анализа скрытых закономерностей в данных, поэтому с их помощью устраняется проблема массовой обработки информации в реальном времени.

Кроме того, для использования деревьев решений требуется малый объем информации, поэтому они занимают в памяти очень мало места. После того как необ-

ходимые знания представлены в виде дерева, чаще всего отпадает необходимость хранить выборки данных, применявшиеся для обучения дерева.

Наконец, деревья решений являются относительно гибкими. Они могут использоваться для обработки непрерывных и символических переменных прогнозирования. Для решения задач регрессии или классификации могут также использоваться переменные отклика, благодаря чему появляется возможность без особых сложностей применять деревья решений для самых разнообразных задач.

Однако следует отметить некоторые недостатки. Деревья решений хорошо подходят для пакетной обработки наборов данных. Но после того как возникает необходимость проводить их обучение в оперативном режиме, существующие алгоритмы могут оказаться довольно громоздкими (требующими большого объема памяти и сложными в реализации). С учетом сказанного, создается впечатление, что наиболее простой подход заключается в том, чтобы инкрементно накапливать выборки, а затем применять пакетное обучение, что обычно приводит к созданию деревьев решений, которые обнаруживают такое же поведение, как и после обучения в оперативном режиме.

Применяемый для обучения деревьев решений алгоритм рекурсивного секционирования по своему характеру является жадным. Этот алгоритм показывает относительно высокую производительность, но обычно не позволяет достичь оптимальных результатов с точки зрения качества и размера дерева. Относительное количество сформированных деревьев, в которых проявляется этот недостаток, удивительно велико, хотя достигаемое при этом качество вполне приемлемо с точки зрения разработки игр.

Последним, но не наименее важным недостатком является то, что в процессе обучения может произойти чрезмерно тщательная подгонка. Для устранения этого недостатка может использоваться еще один этап обработки дерева, на котором осуществляется его усечение, но для этого требуются дополнительные вычисления и дополнительные данные, позволяющие провести аттестацию результатов. Но, к сожалению, интегрированные алгоритмы теряют такое преимущество, как простота.

Резюме

Деревья решений подразделяются на два типа. Применение деревьев классификации приводит к получению категорических ответов, а деревья регрессии возвращают непрерывные значения. Как описано ниже, способ представления деревьев решений является чрезвычайно наглядным и почти одинаковым в обоих случаях.

- В каждом узле принятия решений предусмотрена проверка условия, которое определено на переменных прогнозирования.
- Количество ребер соответствует количеству возможных результатов проверки (взаимоисключающих).
- Уровни дерева образуют иерархическую структуру.
- Листовые узлы возвращают ответы, либо категорические, либо непрерывные.

В алгоритме моделирования используется выборка данных для прохождения по дереву в соответствии с результатами каждой проверки условия. Алгоритм обучения

(или формирования) дерева действует по принципу рекурсивного секционирования, который формулируется, как описано ниже.

- Секционирование набора данных (т.е. разбиение на непересекающиеся подмножества) осуществляется на основе использования наиболее подходящей для этого переменной, что приводит к получению подмножеств, содержащих минимальное количество посторонних включений.
- В дереве создается соответствующий узел принятия решений, и процесс продолжается рекурсивно.

Наилучший способ усовершенствования обучения состоит в том, чтобы сделать набор данных наиболее управляемым, используя дополнительные аттестационные и проверочные наборы. В компьютерных играх может также применяться очень эффективный метод, обеспечивающий отсечение частей дерева, но другие методы, такие как конгломерация и усиление, не очень хорошо подходят для использования в разработке игр.

Деревья решений позволяют находить повторяющиеся шаблоны в данных, а также допускают возможность провести обучение в целях распознавания шаблонов, поэтому, как описано в следующей главе, деревья решений могут применяться при выборе оружия. В результате обучения дерево решений приобретает способность оценивать пригодность каждого вида оружия с учетом ситуации.

Практическая демонстрационная версия

В основе анимата *Detox* лежит архитектура, при создании которой использовалось дерево решений. Деревья решений управляют всеми действиями, которые рассматривались до сих пор в данной книге (например, движением, прицеливанием и выбором оружия), а также соответствующими средствами получения информации о среде. Анимат *Detox* обучается по принципу имитации, собирая данные о действиях людей-игроков. Его производительность является относительно низкой, поскольку каждое действие осуществляется в отрыве от другого (иными словами, в нем не предусмотрена декомпозиция действий с учетом различных форм поведения или способностей). Исходный код и демонстрационная программа для этого анимата доступны в оперативном режиме по адресу `http://AiGameDev.com/`.