Building Classification Models: ID3 and C4.5

Introduction

ID3 and C4.5 are algorithms introduced by Quinlan for inducing *Classification Models*, also called *Decision Trees*, from data.
We are given a set of records. Each record has the same structure, consisting of a number of attribute/value pairs. One of these attributes represents the *category* of the record. The problem is to determine a decision tree that on the basis of answers to questions about the non-category attributes predicts correctly the value of the category attribute. Usually the category attribute takes only the values {*true, false*}, or {*success, failure*}, or something equivalent. In any case, one of its values will mean failure.

For example, we may have the results of measurements taken by experts on some widgets. For each widget we know what is the value for each measurement and what was decided, if to pass, scrap, or repair it. That is, we have a record with as non categorical attributes the measurements, and as categorical attribute the disposition for the widget.

Here is a more detailed example. We are dealing with records reporting on weather conditions for playing golf. The categorical attribute specifies whether or not to Play. The non-categorical attributes are:

```
ATTRIBUTE   |   POSSIBLE VALUES
============+======================
outlook     | sunny, overcast, rain
------------+----------------------
temperature | continuous
------------+----------------------
humidity    | continuous
------------+----------------------
windy       | true, false
============+======================
```

and the training data is:

```
OUTLOOK | TEMPERATURE | HUMIDITY | WINDY | PLAY
=================================================
sunny   |      85     |    85    | false | Don't Play
sunny   |      80     |    90    | true  | Don't Play
overcast|      83     |    78    | false | Play
rain    |      70     |    96    | false | Play
rain    |      68     |    80    | false | Play
rain    |      65     |    70    | true  | Don't Play
overcast|      64     |    65    | true  | Play
sunny   |      72     |    95    | false | Don't Play
sunny   |      69     |    70    | false | Play
rain    |      75     |    80    | false | Play
sunny   |      75     |    70    | true  | Play
overcast|      72     |    90    | true  | Play
overcast|      81     |    75    | false | Play
```

```
rain    |      71     |    80    | true  | Don't Play
```

Notice that in this example two of the attributes have continuous ranges, Temperature and Humidity. ID3 does not directly deal with such cases, though below we examine how it can be extended to do so. A decision tree is important not because it summarizes what we know, i.e. the *training set*, but because we hope it will **classify correctly** new cases. Thus when building classification models one should have both *training data* to build the model and *test data* to verify how well it actually works.

A simpler example from the stock market involving only discrete ranges has Profit as categorical attribute, with values {up, down}. Its non categorical attributes are:

```
ATTRIBUTE   |   POSSIBLE VALUES
============+======================
age         | old, midlife, new
------------+----------------------
competition | no, yes
------------+----------------------
type        | software, hardware
------------+----------------------
```

and the training data is:

```
AGE     | COMPETITION | TYPE    | PROFIT
========================================
old     | yes         | swr     | down
--------+-------------+---------+--------
old     | no          | swr     | down
--------+-------------+---------+--------
old     | no          | hwr     | down
--------+-------------+---------+--------
mid     | yes         | swr     | down
--------+-------------+---------+--------
mid     | yes         | hwr     | down
--------+-------------+---------+--------
mid     | no          | hwr     | up
--------+-------------+---------+--------
mid     | no          | swr     | up
--------+-------------+---------+--------
new     | yes         | swr     | up
--------+-------------+---------+--------
new     | no          | hwr     | up
--------+-------------+---------+--------
new     | no          | swr     | up
--------+-------------+---------+--------
```

For a more complex example, here are files that provide records for a series of votes in Congress. The first file describes the **structure** of the records. The second file provides the **Training Set**, and the third the **Test Set**.

The basic ideas behind ID3 are that:

- In the decision tree each node corresponds to a non-categorical attribute and each arc to a possible value of that attribute. A leaf of the tree specifies the expected value of the categorical attribute for the records described by the path from the root to that leaf. [This defines what is a Decision Tree.]
- In the decision tree at each node should be associated the non-categorical attribute which is *most informative* among the attributes not yet considered in the path from the root. [This establishes what is a "Good" decision tree.]
- *Entropy* is used to measure how informative is a node. [This defines what we mean by "Good". By the way, this notion was introduced by Claude Shannon in Information Theory.]

C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, and so on.

Definitions

If there are n equally probable possible messages, then the probability p of each is 1/n and the information conveyed by a message is -log(p) = log(n). *[In what follows all logarithms are in base 2.]* That is, if there are 16 messages, then log(16) = 4 and we need 4 bits to identify each message.

In general, if we are given a probability distribution P = (p1, p2, .., pn) then the *Information conveyed by this distribution*, also called *the Entropy of P*, is:

```
I(P) = -(p1*log(p1) + p2*log(p2) + .. + pn*log(pn))
```

For example, if P is (0.5, 0.5) then I(P) is 1, if P is (0.67, 0.33) then I(P) is 0.92, if P is (1, 0) then I(P) is 0. [Note that the more uniform is the probability distribution, the greater is its information.]

If a set T of records is partitioned into disjoint exhaustive classes C1, C2, .., Ck on the basis of the value of the categorical attribute, then the information needed to identify the class of an element of T is **Info(T)** = I(P), where P is the probability distribution of the partition (C1, C2, .., Ck):

```
P = (|C1|/|T|, |C2|/|T|, ..., |Ck|/|T|)
```

In our golfing example, we have Info(T) = I(9/14, 5/14) = 0.94,
and in our stock market example we have Info(T) = I(5/10,5/10) = 1.0.

If we first partition T on the basis of the value of a non-categorical attribute X into sets T1, T2, .., Tn then the information needed to identify the class of an element of T becomes the weighted average of the information needed to identify the class of an element of Ti, i.e. the weighted average of Info(Ti):

```
                                      |Ti|
Info(X,T) = Sum for i from 1 to n of  ---- * Info(Ti)
                                      |T|
```

In the case of our golfing example, for the attribute Outlook we have

```
Info(Outlook,T) = 5/14*I(2/5,3/5) + 4/14*I(4/4,0) + 5/14*I(3/5,2/5)
                = 0.694
```

Consider the quantity Gain(X,T) defined as

```
Gain(X,T) = Info(T) - Info(X,T)
```

This represents the difference between the *information needed to identify an element of T* and the *information needed to identify an element of T after the value of attribute X has been obtained*, that is, this is *the gain in information due to attribute X*.

In our golfing example, for the Outlook attribute the gain is:

```
Gain(Outlook,T) = Info(T) - Info(Outlook,T) = 0.94 - 0.694 = 0.246.
```

If we instead consider the attribute *Windy*, we find that Info(Windy,T) is 0.892 and Gain(Windy,T) is 0.048. Thus Outlook offers a greater informational gain than Windy.

We can use this notion of **gain** to rank attributes and to build decision trees where at each node is located the attribute with greatest gain among the attributes not yet considered in the path from the root.

The intent of this ordering are twofold:

- To create small decision trees so that records can be identified after only a few questions.

- To match a hoped for minimality of the process represented by the records being considered(Occam's Razor).

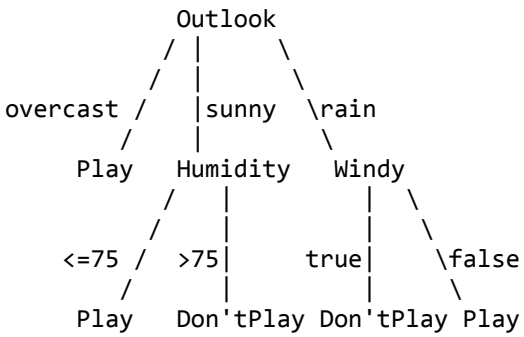
The ID3 Algorithm

The ID3 algorithm is used to build a decision tree, given a set of non-categorical attributes C1, C2, .., Cn, the categorical attribute C, and a training set T of records.

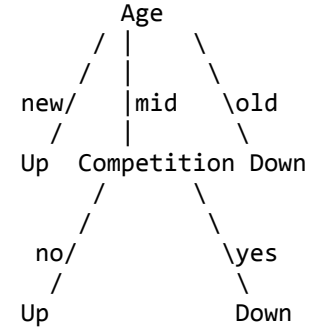
```
function ID3 (R: a set of non-categorical attributes,
              C: the categorical attribute,
              S: a training set) returns a decision tree;
begin
     If S is empty, return a single node with value Failure;
     If S consists of records all with the same value for
        the categorical attribute,
        return a single node with that value;
     If R is empty, then return a single node with as value
        the most frequent of the values of the categorical attribute
        that are found in records of S; [note that then there
        will be errors, that is, records that will be improperly
        classified];
     Let D be the attribute with largest Gain(D,S)
        among attributes in R;
     Let {dj| j=1,2, .., m} be the values of attribute D;
     Let {Sj| j=1,2, .., m} be the subsets of S consisting
        respectively of records with value dj for attribute D;
     Return a tree with root labeled D and arcs labeled
        d1, d2, .., dm going respectively to the trees

          ID3(R-{D}, C, S1), ID3(R-{D}, C, S2), .., ID3(R-{D}, C, Sm);
end ID3;
```

In the Golfing example we obtain the following decision tree:



In the stock market case the decision tree is:



Here is the decision tree, just as produced by c4.5, for the voting example introduced earlier.

Using Gain Ratios

The notion of Gain introduced earlier tends to favor attributes that have a large number of values. For example, if we have an attribute D that has a distinct value for each record, then Info(D,T) is 0, thus Gain(D,T) is maximal. To compensate for this Quinlan suggests using the following ratio instead of Gain:

$$GainRatio(D,T) = \frac{Gain(D,T)}{SplitInfo(D,T)}$$

where SplitInfo(D,T) is the information due to the split of T on the basis of the value of the categorical attribute D. Thus SplitInfo(D,T) is

$$I(|T1|/|T|, |T2|/|T|, .., |Tm|/|T|)$$

where {T1, T2, .. Tm} is the partition of T induced by the value of D.

In the case of our golfing example SplitInfo(Outlook,T) is

$$-5/14*\log(5/14) - 4/14*\log(4/14) - 5/14*\log(5/14) = 1.577$$

thus the GainRatio of Outlook is 0.246/1.577 = 0.156. And SplitInfo(Windy,T) is

$$-6/14*\log(6/14) - 8/14*\log(8/14) = 6/14*0.1.222 + 8/14*0.807 = 0.985$$

thus the GainRatio of Windy is 0.048/0.985 = 0.049

You can run PAIL to see how ID3 generates the decision tree [you need to have an X-server and to allow access (xhost) from yoda.cis.temple.edu].

C4.5 Extensions

C4.5 introduces a number of extensions of the original ID3 algorithm.

In building a decision tree we can deal with training sets that have records with unknown attribute values by evaluating the gain, or the gain ratio, for an attribute by considering only the records where that attribute is defined.

In using a decision tree, we can classify records that have unknown attribute values by estimating the probability of the various possible results. In our golfing example, if we are given a new record for which the outlook is sunny and the humidity is unknown, we proceed as follows:

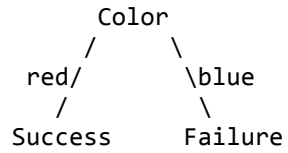
We move from the Outlook root node to the Humidity node following the arc labeled 'sunny'. At that point since we do not know the value of Humidity we observe that if the humidity is at most 75 there are two records where one plays, and if the humidity is over 75 there are three records where one does not play. Thus one can give as answer for the record the probabilities (0.4, 0.6) to play or not to play.

We can deal with the case of attributes with **continuous ranges** as follows. Say that attribute Ci has a continuous range. We examine the values for this attribute in the training set. Say they are, in increasing order, A1, A2, .., Am. Then for each value Aj, j=1,2,..m, we partition the records into those that have Ci values up to and including Aj, and those that have values greater than Aj. For each of these partitions we compute the gain, or gain ratio, and choose the partition that maximizes the gain.
In our Golfing example, for humidity, if T is the training set, we determine the information for each partition and find the best partition at 75. Then the range for this attribute becomes {<=75, >75}. Notice that this method involves a substantial number of computations.

Pruning Decision Trees and Deriving Rule Sets

The decision tree built using the training set, because of the way it was built, deals correctly with most of the records in the training set. In fact, in order to do so, it may become quite complex, with long and very uneven paths.

Pruning of the decision tree is done by replacing a whole subtree by a leaf node. The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf. For example, if the simple decision tree



is obtained with one training red success record and two training blue Failures, and then in the Test set we find three red failures and one blue success, we might consider replacing this subtree by a single Failure node. After replacement we will have only two errors instead of five failures.

Winston shows how to use *Fisher's exact test* to determine if the category attribute is truly dependent on a non-categorical attribute. If it is not, then the non-categorical attribute need not appear in the current path of the decision tree.

Quinlan and Breiman suggest more sophisticated pruning heuristics.

It is easy to *derive a rule set from a decision tree*: write a rule for each path in the decision tree from the root to a leaf. In that rule the left-hand side is easily built from the label of the nodes and the labels of the arcs.

The resulting rules set can be simplified:

Let LHS be the left hand side of a rule. Let LHS' be obtained from LHS by eliminating some of its conditions. We can certainly replace LHS by LHS' in this rule if the subsets of the training set that satisfy respectively LHS and LHS' are equal.

A rule may be eliminated by using metaconditions such as *"if no other rule applies"*.

You can run the C45 program here [you need to have an X-server and to allow access (xhost) from yoda.cis.temple.edu].

Classification Models in the Undergraduate AI Course

It is easy to find implementations of ID3. For example, a Prolog program by Shoham and a nice Pail module.

The software for C4.5 can be obtained with Quinlan's book. A wide variety of training and test data is available, some provided by Quinlan, some at specialized sites such as the University of California at Irvine.

Student projects may involve the implementation of these algorithms. More interesting is for students to collect or find a significant data set, partition it into training and test sets, determine a decision tree, simplify it, determine the corresponding rule set, and simplify the rule set.

The study of methods to evaluate the error performance of a decision tree is probably too advanced for most undergraduate courses.

References

Breiman,Friedman,Olshen,Stone: Classification and Decision Trees
Wadsworth, 1984

A decision science perspective on decision trees.

Quinlan,J.R.: C4.5: Programs for Machine Learning
Morgan Kauffman, 1993

Quinlan is a very readable, thorough book, with actual usable programs
that are available on the internet. Also available are a number of
interesting data sets.

Quinlan,J.R.: Simplifying decision trees
International Journal of Man-Machine Studies, 27, 221-234, 1987

Winston,P.H.: Artificial Intelligence, Third Edition
Addison-Wesley, 1992

Excellent introduction to ID3 and its use in building decision trees and,
from them, rule sets.

---

ingargiola@cis.temple.edu