Some Experiments in Decision Tree Induction

G.J. Williams†

This paper presents a number of experiments dealing with various aspects of the ID3 decision tree induction algorithm. The aspects dealt with include the choice of attribute to split on, exception split pruning, and combining decision trees.

Keywords and Phrases: **Decision tree induction, knowledge acquisition, machine learning, geographic problem solving, classification problem, ID3.**
CR Categories: **I.2.6, I.2.1**

INTRODUCTION

A number of algorithms have been developed to construct decision trees from examples of decisions made by an expert. Such algorithms employ a 'divide and conquer' technique, repeatedly dividing the collection of examples into smaller and more homogeneous collections. The divides correspond to branching in the decision tree, and each node of the tree corresponds to a collection of examples to be conquered.

Some success in using these techniques to aid in the induction of knowledge bases for use in knowledge-based systems has been achieved (Quinlan *et al.*, 1986). However, there are a number of problems with such induction techniques, and research aimed at improving them is continuing.

This paper presents a number of experiments carried out in order to investigate various aspects of Quinlan's ID3 (Quinlan, 1986) decision tree induction algorithm (described in the next section). Various modifications and enhancements are made to the basic algorithm, each then being used in a real-world example to induce decision trees. The modifications made include changes to bias, categorisation of integer attributes, and exception split pruning. The resulting decision trees are applied to a large database dealing with geographic information, classifying each of the database objects (i.e. records). These results are compared to results produced by a regression model, which was constructed from a specially chosen test set of objects from the same database. Comparisons are also made between the modified decision tree algorithm and the ID3 algorithm.

THE DECISION TREE ALGORITHM

The decision tree induction algorithm used for these experiments involves four basic steps, in common with other induction algorithms.

The induction begins with T, a *training set* consisting of a set of examples of a *domain expert's* decisions. Each example is thought of as an object described by a number of attributes, each object having been classified by the domain expert, representing his/her decision about that object.

The ID3 algorithm allows attributes of two types: those whose values are taken from a small finite unordered set of possible values (*categorical attributes*) and those whose values are integers (*integer attributes*). An example of an object, identified as 'Region 19481' is:

Region 19481: Soil is of type CC1,
Distance to nearest seaport is 836 km,
Average weekly moisture index for winter is 21%.

The soil type (a categorical attribute), the distance to nearest seaport (an integer attribute), and the average weekly moisture index for winter (an integer attribute) have the values CC1, 836, and 21 respectively.

The algorithm first considers a number of partitions of the training set, T. This set of alternative partitions is denoted by **S**, with the member partitions identified as S_i, $i = 1,\ldots,n$. Each partition consists of a number of cells, each cell containing a collection of objects from T. The cells of a given partition are identified as C_j, $j = 1,\ldots,p$. Each S_i represents an alternative branching pattern from the current node in the decision tree.

ID3 considers only those partitions which divide the objects according to a single attribute. Thus, a partition associated with the attribute Soil will contain a cell for each of the values of Soil in the training set.

†*Department of Computer Science, Australian National University, G.P.O. Box 4, Canberra, Australia 2601.* *This paper was presented at the tenth Australian Computer Science Conference at Deakin University in Geelong, Victoria in February, 1987. Manuscript received March, 1987.*

Members of a cell have the same value for their Soil attribute. Each partition, then, is associated with exactly one attribute, and each categorical attribute is associated with exactly one partition. Integer attributes may define more than one partition, each containing only two cells. One cell consists of those objects with a value for the integer attribute less than some value n, the other cell having those objects whose value for the integer attribute is greater than or equal to that same n.

The next step in the general algorithm is to choose S*, the best partition from **S**. ID3 bases its choice of S* on information theory, computing $E(A)$ as the expected information required for the tree with attribute A as the root (Quinlan, 1986). The attribute with the minimum value for $E(A)$ is chosen as the root of the decision tree. $E(A)$ can be expressed in terms of p_{ij}, where p_{ij} is the number of objects in C_i which are classified as class j. For categorical attributes with n values,

$$E(A) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \ln \frac{p_{ij}}{\sum_{k=1}^{m} p_{ik}}$$

where m is the number of classes. For integer attributes, n is taken as two, and of all possible binary splits of the training set with respect to the integer attribute, the one with the minimum value of $E(A)$ is chosen as the value of $E(A)$ for that attribute.

The third step of the algorithm involves constructing a *discriminating description* of each cell of S*. Such a description of a cell describes each object in that cell, but does not describe any object in any other cell of S*. These discriminating descriptions are the branch labels in the decision tree.

Branch labelling in ID3 is trivial. Since the node from which the branches emanate is labelled with a single attribute, the branches are labelled with the various values of the attribute (for categorical attributes) or with '$> n$' and '$\leq n$' (for integer attributes), where n is the midpoint of the binary split on the integer attribute.

The final step of the algorithm tests a stopping criterion to determine if the dividing and conquering can stop. If the criterion is not met, then the cells of S* are each used as new training sets, and the algorithm is re-applied to each. The stopping criterion used in ID3 tests for class homogeneous cells.

The general algorithm is summarised as:

1. Construct **S**, the set of alternative partitions of the training set.
2. Select the best S* $\in$ **S**.
3. Find a discriminating description for each cell, C_i, in S*.
4. For each C_i, test the stopping criterion. If it is not met, then use C_i as a training set and repeat from step 1.

AN EXAMPLE: THE ARID DATABASE

The Database

All the experiments to be described below use the ARID database. This is a subset of the Australian Resources Information System (ARIS) which is a continental-scale geographic information system (Walker, Cocks & Young, 1985). The ARID database consists of objects corresponding to single grid-cells, which are approximately 700 square kilometre rectangular regions. Australia divides into 11, 109 such regions, 8413 of which are arid regions and form the ARID database. For each object, values of some 40 attributes are maintained, including, for example, the dominant soil type, the distance to the nearest seaport, and a number of moisture indices.

Because of the importance of agriculture to Australia's economy, advice on how best to make use of our vast land area is essential. The ARID database, for example, has been used for the prediction of the viability for the pastoral use of land in the arid regions (Cocks, Young & Walker, 1986). A regression model was constructed based upon 106 representative objects from the ARID database. The objects in this training set, referred to as the T106 training set, were classified by an agricultural scientist (the domain expert), and used to construct a first approximation model of grazing viability. This model was then refined until it gave predictions for viability which corresponded (88%) to the expert's opinion for all of the objects in the training set. The model was then applied to the rest of the database to provide predictions for viability for the arid regions of Australia. These predictions, on the whole, have been accepted as realistic by the domain expert.

This regression model is used in these experiments as the domain expert. The conclusions of the model are assumed correct, and form the basis of comparisons used to judge the quality of the classifications produced by the decision trees. The same training set, T106, is used for the experiments, but with the classifications given to its members by the Model rather than the original domain expert.

The attributes selected by the expert as being relevant to the problem of deciding viability for pastoral use included the predominant soil category (Soil), the predominant class of upper and lower vegetation (UVeg and LVeg), the distance in kilometres to the nearest seaport (DPort), and three moisture indicators (AWMIH, AWMIS, and AWMIW). AWMIH is the average weekly moisture index for the wettest consecutive 13 weeks of the year, AWMIS is the average weekly moisture index from November to April inclusive (i.e. Summer), and AWMIW is the average weekly moisture index from May to October inclusive (i.e. Winter). The Soil attribute has 30 possible values, whilst UVeg and LVeg have 50 and 41 possible values respectively. The moisture indices take integer

values in the range 0 to 100. The classes are referred to as VLow, Low, Medium and High in these experiments.

The T106 Training Set

The composition of the objects in the T106 training set is summarised below.

Of the 30 possible values for Soil, the training set contains examples of only 9. There are only 17 of the possible 50 values for UVeg represented in T106, and only 15 of the possible 41 values for LVeg.

The values for DPort in T106 range from 121 km to 1225 km, there being 99 distinct values. AWMIH ranges ranges from nine to 95, with 46 distinct values in this range. Similarly the AWMIS attribute ranges from eight to 83 with 30 distinct values, and AWMIW ranges from four to 43, with 25 distinct values.

The class associated with each object, which indicates the viability for pastoralism for the corresponding region, is represented by 14 objects classified as VLow, 16 as Low, 40 as Medium and 36 as High in T106.

Constructing A Decision Tree: T106DC

An application of the ID3 algorithm to the T106 training set was carried out. The values of the ID3 cost, $E(A)$, associated with each of the attributes when choosing the initial root of the decision tree, is given in Table 1 below. Note that the value of $E(A)$ for the integer attributes is the best obtainable by any binary split on the integers. For these best binary splits the split point is also given below.

A	E(A)	Split
Soil	62.94	
UVeg	68.62	
LVeg	77.69	
DPort	102.80	925
AWMIH	116.24	20
AWMIS	119.83	31
AWMIW	105.62	11

Table 1. Attribute costs for root of T106DC.

The complete decision tree constructed is shown in Figure 1. The attribute Soil, having the minimum value of $E(A)$ (see Table 1), labels the root of the decision tree. There are nine branches leading from this root; the branches labelled with 2 and 12 are combined, since they lead to the same subtree. Each branch corresponds to one of the nine values for the Soil attribute found in the T106 training set. For those values of Soil which do not appear, there exists an implicit branch leading to a subtree consisting of just the node Null, indicating that no class can be assigned.

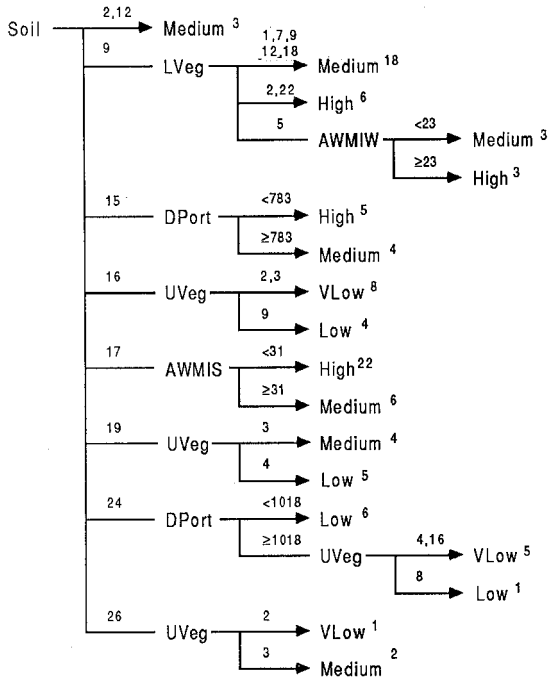

Figure 1: Decision tree T106DC

Each leaf node of the decision tree is labelled with a decision, i.e. class, along with a superscript which is the number of objects in the training set corresponding to this leaf node. For example, there are three objects in the T106 training set which have a value of 2 or 12 for the Soil attribute, each of which is classified as Medium.

The decision tree so constructed, T106DC, can now be applied to all of the objects in the ARID database. A summary of the application of both the T106DC decision tree and the Model to the ARID database is given in Table 2. It is noted that whilst the Model is able to classify (or *cover*) all objects, T106DC only covers 5924 of them. That is, there exists objects in the database for which there is no corresponding path through the decision tree from the root to a leaf node. In decision tree T106DC, for example, any object with Soil = 10 can not be classified (and so is not covered).

Class:	VLow	Low	Medium	High	Total
Model:	1691 (20.1%)	2484 (29.5%)	2763 (32.8%)	1475 (17.5%)	8413
T106DC:	632 (10.7%)	1879 (31.7%)	2224 (37.5%)	1192 (20.1%)	5924

Table 2. Model and T106DC applied to ARID.

A comparison of the answers given by the Model and by T106DC is presented in Table 3. It is seen from the table that the decision tree agrees with the model 71.5% of the time (excluding those objects not covered by T106DC). Three degrees of disagreement are introduced. Since it is known that an ordering

exists on the classes, VLow < Low < Medium < High, we say that two classifications of one object mildly disagree if the two classes are neighbours in this ordering. A moderate disagreement occurs when the two classes are separated by one class, and a strong disagreement occurs when the two are separated by two classes. Strong disagreements can occur only between VLow and High. From the table it is seen that the Model and T106DC agree or only mildly disagree on 98.2% of the data base.

Agree	Mildly Disagree	Moderately Disagree	Strongly Disagree
4327	1581	106	0
(71.5%)	(26.7%)	(1.8%)	0

Table 3. Comparing T106DC to the Model.

THE EXPERIMENTS

A number of aspects concerning the ID3 algorithm, as well as other decision tree constructing algorithms, have often been left unspecified, or have not been dealt with. Two aspects dealt with in these experiments centre upon the sensitivity of the cost function and the problem of generalising overly specific decision trees.

The first issue relates to the sensitivity of the ID3 cost function, $E(A)$, to the data. A brief scan of Table 1, for example, shows that the three categorical attributes have roughly the same value, and that the four integer attributes likewise have similar values. Two points are made from this observation. Firstly, how should the 'best' attribute be chosen from amongst a number of approximately equal best attributes, especially if the cost function returns values which have some uncertainty about them, due to uncertainty about the data? Secondly, does the type of the attribute bias the cost computed for that attribute? These two aspects of the cost function are considered first.

T106DI: Conflict Handling

This first experiment is concerned with the problem of deciding which of a number of equally good attributes, to choose. On several occasions in constructing T106DC, for example, both UVeg and DPort had equal values computed for their costs. Because of an implicit ordering on the attributes in the ID3 algorithm, UVeg was always chosen. However, the choice of DPort would seem just as good from the point of view of the cost function. If the cost function can be relied upon, then it should be expected that other choices lead to decision trees which are just as good.

Decision tree T106DI was constructed by choosing integer attributes over categorical attributes, whenever their costs were the same. The specified attribute ordering was

{DPort AWMIH AWMIS AWMIW Soil UVeg LVeg}

compared to an ordering of

{Soil UVeg LVeg DPort AWMIH AWMIS AWMIW}

used in constructing T106DC.

The resulting decision tree again has Soil as the root, as there is no equal competitor for this position. The tree differs from T106DC in only three of the nine branches emanating from this root. (See Figure 2.) Of these, two represent changes of choice from the categorical attribute UVeg to the integer attribute DPort, and the other from UVeg to AWMIH.

Table 4 below indicates that T106DI did not perform as well as T106DC. Because integer attributes are preferred to categorical attributes, and branches from integer attributes cover all possible values of that attribute, the coverage was expected to increase. Although the coverage by T106DI is approximately 20% greater than that of T106DC, the classifications given by T106DI to the extra objects mostly disagree with the classifications given to them by the Model. Of the 1164 extra objects covered, only 30.8% are classified in agreement with the Model. There are no strong disagreements, and of the new disagreements, only 5% are moderate. The combined agreement/mild disagreement percentage is 98%.

Application	Agree	Disagree	Coverage
T106DC:	4237	1687	5924
	(71.5%)	(28.5%)	(70.4%)
T106DI:	4595	2493	7088
	(64.8%)	(35.2%)	(84.3%)

Table 4. T106DC and T106DI compared to the Model.

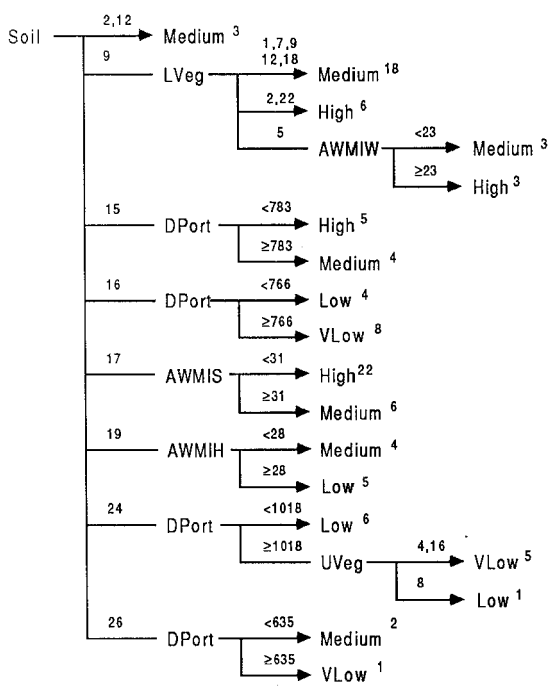

Figure 2: Decision tree T106DI

It is seen then that significantly different results are obtained depending upon which equally best attribute is chosen. In this experiment it is also seen that choosing integer attributes over categorical attributes results in decision trees with greater coverage, as would be expected. With this greater coverage though comes the cost of a slight decrease in the accuracy of the decision tree.

T106Aa: Categorising Integer Attributes

This second experiment deals with the problem of bias in the ID3 cost function. Quinlan (1986, p. 100) has shown that the cost function favours categorical attributes with many values. Since integer attributes are handled as binary valued categorical attributes, and the categorical attributes used in these experiments have many values, there is a strong bias against integer attributes. In an attempt to alleviate this problem, the AW attributes (AWMIH, AWMIS, and AWMIW), are treated, in the following experiments, as categorical attributes. As has been previously noted, the ARID training set contains objects which have 46 distinct values for AWMIH, 30 distinct values for AWMIS, and 25 for AWMIW. In these experiments, the distinct values become the categories of each of the respective attributes.

An integer attribute should only be considered as a categorical attribute when the number of distinct values of that attribute occurring in the training set is much less than the size of the training set. Such a restriction avoids decision tress which have large branching factors, a result of having a branch for each of the possible values of the attribute. The integer attribute DPort is thus ruled out for these experiments. DPort could be included though by categorising it into 'n' subranges, where n might be the average number of categories in the other categorical attributes. Such a categorisation is not considered here.

The decision tree T106Aa was constructed using the ID3 cost function for the choice of attribute at each node. An ordering of

{Soil UVeg LVeg AWMIH AWMIS AWMIW DPort}

was used whenever there was a tie for the minimum value of $E(A)$.

It is expected that the resulting decision tree will not cover as much of the ARID database as previous trees have, since only a limited number of all the possible values of the AW attributes are now ever considered. Also, the chances of selecting one of the AW attributes for the root of any tree is increased. The resulting decision tree is illustrated in Figure 3.

The above expectations are borne out. The initial root node of T106Aa is AWMIW, with an ID3 cost of 59.75. This compares with a value of 105.62 when AWMIW is considered as an integer attribute in T106DC. The complete list of values for the ID3 costs

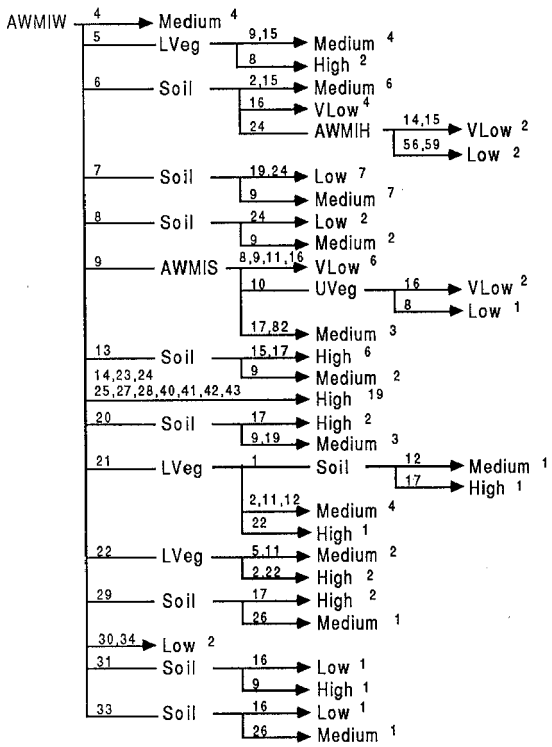

Figure 3: Decision tree T106Aa.

in choosing the initial root of the T106Aa decision tree is given in Table 5.

Applying the decision tree to the ARID database results in a coverage by T106Aa of only 2963 objects (35.2%). Of this coverage, the ratio of agreement to disagreement with the Model is approximately 1:1. However an analysis of the disagreements is more encouraging. Of the three degrees of disagreement introduced previously, 59% of the disagreements are mild (e.g. classified by the Model as High, but as Medium by T106Aa), whilst 36% are moderate (e.g. classified by the Model as High, but as Low by T106Aa), and only 5% are strong (e.g. classified as High by the Model, but as VLow by T106Aa).

A	E(A)	Split
Soil	62.94	
UVeg	68.62	
LVeg	77.69	
DPort	102.80	925
AWMIH	60.91	
AWMIS	82.62	
AWMIW	59.75	

Table 5. Attribute costs for root of T106Aa.

A number of modifications are made below in an attempt to improve the performance of this decision tree.

T106Ad: Splitting Continuous Attributes

One technique for generalising the decision tree is to make use of the knowledge that the AW attributes are really continuous. It then makes sense, where appropriate, to introduce midpoint, or multiway, splits, similar to ID3. An obvious example in T106Aa is the subtree whose root is along the path with AWMIW = 6, Soil = 24 (refer to Figure 3). This root is labelled with AWMIH, and has two branches. One branch is labelled with '14, 15', and leads to VLow, and the other branch is labelled with '56, 59', and leads to Low. There appears to be a very strong case for introducing a binary split here.

T106Ad, then, is the same as T106Aa with the above subtree rooted at AWMIH replaced with a binary split subtree with the same root, and a split point of 35. For this path, objects with a value for AWMIH less than 35 are classed as VLow, whereas objects with values greater than or equal to 35 are classed as Low. With such a modification the only change that can occur to the resulting classifications is that some objects which were classified as Null by T106Aa can now be classified as either VLow or Low.

Applying T106Ad to the ARID database results in 303 fewer Nulls. Of these, 69% agree with the Model, and 31% mildly disagree with the Model.

T106Af: A Further Split-Generalisation

Another obvious candidate for split is the subtree rooted with AWMIS on the path AWMIW = 9. If the AWMIS = 10 branch of this subtree is ignored, then a binary split can be made with a split point of 17. Thence, for AWMIS less than 17, the class is VLow, and for AWMIS greater than or equal to 17 the class is Medium. For this experiment, the exception when AWMIS = 10 is kept, so that an object is classified using the binary split only if AWMIS $\neq$ 10.

Making this modification to T106Ad increases the coverage of the ARID database by another 256 objects of which 32% agree with the Model. Together, agreement and mild disagreement account for 78% of the increased coverage.

T106Ae: Removing an Exception Split

The part of the T106Aa decision tree modified in T106Af contains an *exception split*. An exception split arises arises when most of the members of a training set belong to a single class, but the training set contains some objects not in that class — the exceptions. If the exceptions account for only a small percentage of the whole training set, then there is a possibility that they are due to noise in the training set, resulting in them being misclassified. (Quinlan (1986) discusses the problem of noise.)

One approach to handling exceptions splits is to reclassify them in agreement with the majority of objects in the training set containing the exceptions. Decision tree T106Ae was constructed by reclassifying the single exception in the training set with AWMIW = 9, and AWMIS < 17 from Low to VLow, in agreement with the other 8 members of this training set. T106Ae modifies T106Af by removing the appropriate subtree. Thus T106Ae no longer correctly classifies the T106 training set, being 0.94% in disagreement.

Applying T106Ae to the ARID database results in only 60 more AIRD objects being covered. Of these, only 23 agree with the Model, although a further 22 are only mild disagreements. Of the other 15, four strongly disagree and 11 moderately disagree. These poor results indicate that the exception should probably be kept in this case.

T106Ag: Changing The Root

The performance, with respect to the Model, of the T106A decision trees constructed so far has been somewhat poor when compared to the performance of T106DC. As already mentioned, the poor coverage is due, to a large degree, to the choice of AWMIW as the root of the decision tree. However the higher percentage of disagreement is also of concern. In an attempt to improve the performance of these decision trees, and to further investigate the technique of categorising integer attributes, decision tree T106Ag was constructed.

The ID3 algorithm was again employed, as in T106Aa, but now the choice for the root node of the decision tree was overridden. Recalling that ID3 selects AWMIW as the root of T106Aa, here Soil is selected as the root because that choice by ID3 in T106DC lead to a good decision tree. All other choices of attributes for T106Ag are left to ID3. The resulting decision tree is given in Figure 4.

The performance of the resulting decision tree is indeed markedly improved. The coverage, being 4577 ARID objects, is significantly better than that of the other T106A experiments discussed above. Of all the T106A experiments, it also has the greatest percentage agreement (67.9%) with the Model. It still does not reach T106DC's level of performance though.

T106Aj: Split-Generalisations

A number of generalisations using our knowledge of the true nature of the AW attributes are possible. These are for the paths Soil = 9, Soil = 15, and Soil = 17. For the first of these, there are two clear groupings, with some exceptions around the split point where AWMIW = 21, 22 (refer to Figure 4). Drawing upon the experience of removing the exceptions in T106Ae, the exceptions around the split point are kept. The second candidate for generalisation, on the path Soil = 15, has a boundary exception split, and although a three way split may be feasible here, a two-way split is made, with the case AWMIH = 10 as an exception. The third generalisation is straight forward, with a split point of AWMIH = 19, and results in the construction of the same subtree as found in T106DC.

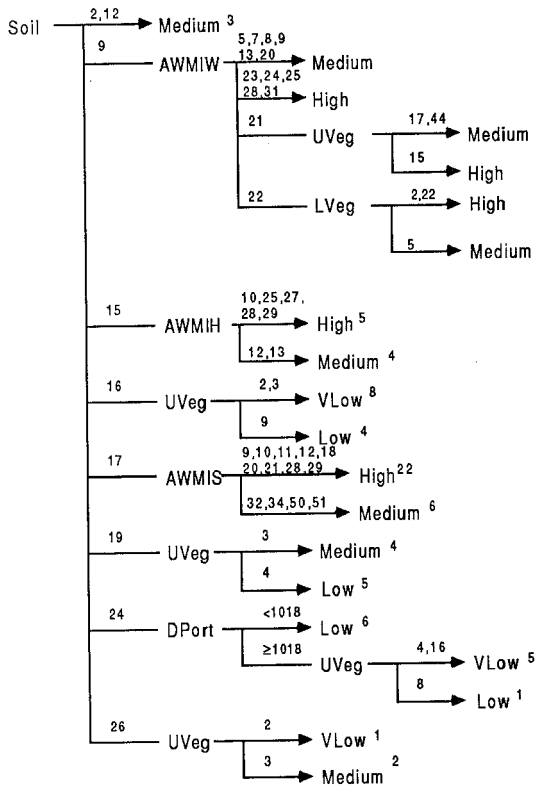

Figure 4: Decision tree T106Ag.

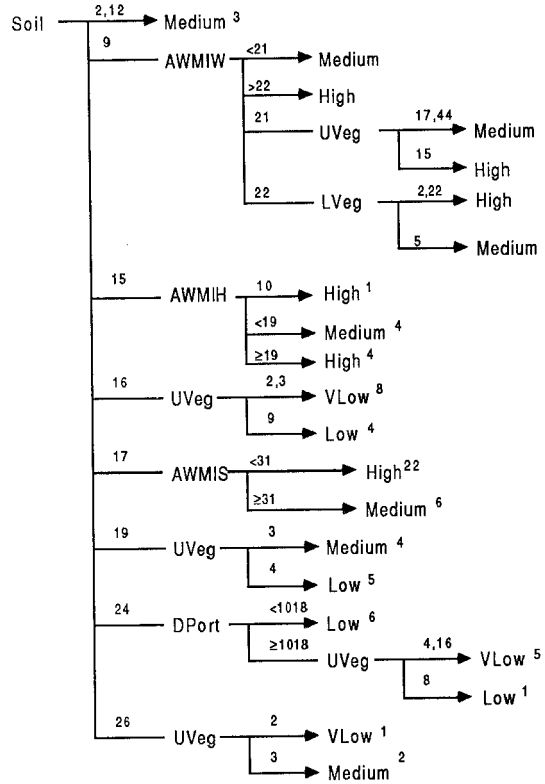

Figure 5: Decision tree T106Aj.

Each of these generalisations in turn improves the decision tree. The resulting decision tree, T106Aj of Figure 5, is almost identical to T106DC, but improves upon T106DC in its coverage (by 7.5%), without losing its accuracy (71.3% agreement).

SUMMARY

Table 6 presents a summary of the experiments described in this paper.

This paper has presented a collection of techniques for modifying decision tree building algorithms. The techniques introduced include the categorisation of integer attributes, split-generalisations on such attributes, and exception split handling. It has also been demonstrated that choosing different equally good attributes can lead to significant changes to the quality of the resulting decision tree.

For the purpose of these experiments, a regression model was used as the domain expert. Thus, classifications given to objects by the regression model were used to assess the quality of the decision trees by comparing the Model's classifications with the classifications given by the decision tree.

Experiment T106DI shows that where contention arises over the choice of attribute, then a strategy of favouring integer attributes over categorical attributes results in decision trees with wider coverage. In the case of T106DI, the accuracy of the decision tree has suffered to a small degree, but not significantly. Similar results would be expected in the general case, since integer splits cover all possible values of the integer, whereas categorical splits only cover those values found in the training set.

A problem with the ID3 cost function is that it has a bias toward many valued categorical attributes. The approach explored in experiment T106Aa of categorising certain integer attributes did not initially prove to produce good results, due to a dramatic decrease in the coverage of the decision tree. Coupled with this was a significant increase in the moderate and strong degrees of disagreement. However, this approach, coupled with suitable generalisation heuristics does offer some promise.

Generalising the pseudo-categorical attributes in the decision tree is shown to be of benefit in the T106Ad and T106Af experiments. Both of these experiments result in improvements in coverage, along with increases in agreement.

The process of removing exception splits, as illustrated in experiment T106Ae, did not result in any significant changes, but further experimentation is required.

The T106Ag and T106Aj experiments showed that restricting the root node of the decision tree to be a categorical attribute rather than a pseudo-categorical attribute, but then allowing pseudo-categorical attributes elsewhere, resulted in a decision tree with better coverage, and better accuracy. Combining this with

Decision Tree	Description	Coverage	Agree	Mild	Moderate	Strong
T106DC	Application of ID3.	70.8	71.5	26.7	1.8	0
T106DI	Favour integer attributes when equal minimum cost occurs.	84.3	64.8	33.1	2.1	0
T106Aa	Application of ID3 with the AW attributes as categorical.	35.2	48.4	30.6	18.6	2.3
T106Ad	Split generalisation of integer attributes.	38.8	50.3	30.7	16.9	2.1
T106Af	Further split generalisation.	41.9	49.0	31.8	17.0	2.3
T106Ae	Remove exception split.	42.8	48.8	31.6	17.0	2.3
T106Ag	Categorical attribute as root.	54.4	67.9	29.8	2.3	0
T106Aj	Split generalisation in T106Ag	75.7	71.3	27.1	1.7	0

Table 6. Summary of decision tree applications, with comparisons to the Mode — all figures are percentages.

the generalisation techniques described above, produces a decision tree of greater coverage and greater accuracy than ID3 alone (T106DC).

Before these results can be generalised to any decision tree construction task, further experimentation, on other tasks (i.e. databases) is required. However, a number of other experiments have been carried out on the ARID database using different training sets, of differing sizes. The preliminary results from these further experiments have agreed, in spirit, with those presented here.

ACKNOWLEDGEMENTS

The database used here was made available to me by Paul Walker (CSIRO Division of Water and Land Resources). Thanks also belong to Robin Stanton and Malcolm Newey (Computer Science, ANU) and to Richard Davis (CSIRO Division of Water and Land Resources) for supervising this work and for reading drafts of this paper.

References

COCKS, K. D., YOUNG, M. D., & WALKER, P. A. (1986): Mapping relative viability prospects for pastoralism in Australia, *Agricultural Systems*, Vol. 20, pp. 175-193.

QUINLAN, J. R. (1986): Induction of Decision Trees, *Machine Learning*, Vol. 1, No. 1, pp. 81-106.

QUINLAN, J. R., COMPTON, P. J., HORN, K. A., & LAZARUS, L. (1986): 'Inductive Knowledge Acquisition: A Case Study', in *Proceedings of the Second Australian Conference on Applications of Expert Systems*, New South Wales Institute of Technology.

WALKER, P. A., COCKS, K. D., & YOUNG, M. D. (1985): Regionalising continental data sets, *Cartography*, Vol. 14, No. 1, pp. 66-73.

Biographical Note

Graham Williams is a doctoral research student in Computer Science at the Australian National University. He received his B.Sc. (with honours) from the University of Adelaide in 1983. His research focusses on the acquisition and representation of knowledge for use in expert systems. He developed a successful expert system for fire management whilst on leave from his doctoral studies with the CSIRO Division of Water and Land Resources. Mr Williams is also an AI consultant to BBJ Computers International in Melbourne, where he will shortly take up a position as Research and Development Manager.