An Implementation of ID3 --- Decision Tree Learning Algorithm

Wei Peng, Juhua Chen and Haiping Zhou
Project of Comp 9417: Machine Learning
University of New South Wales, School of Computer Science & Engineering,
Sydney, NSW 2032, Australia
weipengtiger@hotmail.com

Abstract

Decision tree learning algorithm has been successfully used in expert systems in capturing knowledge. The main task performed in these systems is using inductive methods to the given values of attributes of an unknown object to determine appropriate classification according to decision tree rules. We examine the decision tree learning algorithm ID3 and implement this algorithm using Java programming. We first implement basic ID3 in which we dealt with the target function that has discrete output values. We also extend the domain of ID3 to real-valued output, such as numeric data and discrete outcome rather than simply Boolean value. The Java applet provided at last section offers a simulation of decision-tree learning algorithm in various situations. Some shortcomings are discussed in this project as well.

1 Introduction

1.1 What is Decision Tree?

What is decision tree: A decision tree is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node represents a decision.

Decision tree are commonly used for gaining information for the purpose of decision -making. Decision tree starts with a root node on which it is for users to take actions. From this node, users split each node recursively according to decision tree learning algorithm. The final result is a decision tree in which each branch represents a possible scenario of decision and its outcome.

1.2 What is decision tree learning algorithm?

'Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. Decision tree learning is one of the most widely used and practical methods for inductive inference'. (Tom M. Mitchell,1997,p52)

Decision tree learning algorithm has been successfully used in expert systems in capturing knowledge. The main task performed in these systems is using inductive methods to the given values of attributes of an unknown object to determine appropriate classification according to decision tree rules.

Decision trees classify instances by traverse from root node to leaf node. We start from root node of decision tree, testing the attribute specified by this node, then moving down the tree branch according to the attribute value in the given set. This process is the repeated at the sub-tree level.

What is decision tree learning algorithm suited for:

1. Instance is represented as attribute-value pairs. For example, attribute 'Temperature' and its value 'hot', 'mild', 'cool'. We are also concerning to extend attribute-value to continuous-valued data (numeric attribute value) in our project.

2.The target function has discrete output values. It can easily deal with instance which is assigned to a boolean decision, such as 'true' and 'false', 'p(positive)' and 'n(negative)'. Although it is possible to extend target to real-valued outputs, we will cover the issue in the later part of this report.

3.The training data may contain errors. This can be dealt with pruning techniques that we will not cover here.

The 3 widely used decision tree learning algorithms are: ID3, ASSISTANT and C4.5. We will cover ID3 in this report.

1.3 Why is Decision Tree Learning an attractive Inductive Learning method

> 'Purely inductive learning methods formulate general hypotheses by finding empirical regularities over the training examples.'
>
> (Tom M. Mitchell,1997,p334)

For inductive learning, decision tree learning is attractive for 3 reasons:

1. Decision tree is a good generalization for unobserved instance, only if the instances are described in terms of features that are correlated with the target concept.
2. The methods are efficient in computation that is proportional to the number of observed training instances.
3. The resulting decision tree provides a representation of the concept that appeals to human because it renders the classification process self-evident.

(Paul Utgoff & Carla Brodley, 1990)

2 Decision Tree Learning Algorithm — ID3 Basic

2.1. ID3 Basic

ID3 is a simple decision tree learning algorithm developed by Ross Quinlan (1983). The basic idea of ID3 algorithm is to construct the decision tree by employing a top-down, greedy search through the given sets to test each attribute at every tree node. In order to select the attribute that is most useful for classifying a given sets, we introduce a metric---information gain.

To find an optimal way to classify a learning set, what we need to do is to minimize the questions asked (i.e. minimizing the depth of the tree). Thus, we need some function which can measure which questions provide the most balanced splitting. The information gain metric is such a function.

2.2 Entropy --- measuring homogeneity of a learning set

(Tom M. Mitchell,1997,p55)

> In order to define information gain precisely, we need to discuss entropy first.
>
> First, lets assume, without loss of generality, that the resulting decision tree classifies instances into two categories, we'll call them P(positive)and N(negative).

Given a set S, containing these positive and negative targets, the entropy of S related to this boolean classification is:

Entropy(S)=

- P(positive)log_2P(positive) - P(negative)log_2P(negative)

P(positive): proportion of positive examples in S

P(negative): proportion of negative examples in S

For example, if S is (0.5+, 0.5-) then Entropy(S) is 1, if S is (0.67+, 0.33-) then Entropy(S) is 0.92, if P is (1+, 0-) then Entropy(S) is 0. Note that the more uniform is the probability distribution, the greater is its information.

You may notice that entropy is a measure of the impurity in a collection of training sets. But how it is related to the optimisation of our decision making in classifying the instances. What you will see at the following will answer this question.

2.3 Information Gain --- measuring the expected reduction in Entropy

(Tom M. Mitchell,1997,p57)

As we mentioned before, to minimize the decision tree depth, when we traverse the tree path, we need to select the optimal attribute for splitting the tree node, which we can easily imply that the attribute with the most entropy reduction is the best choice.

We define information gain as the expected reduction of entropy related to specified attribute when splitting a decision tree node.

The information gain, Gain(S,A) of an attribute A,

Gain(S,A)= Entropy(S) -

Sum for v from 1 to n of (|Sv|/|S|) * Entropy(Sv)

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 intention of this ordering is:

1. To create small decision trees so that records can be identified after only a few decision tree splitting.

2. To match a hoped for minimalism of the process of decision making

3 What we have done here

3.1 How we implement ID3 Algorithm here:

ID3 (Learning Sets S, Attributes Sets A, Attributesvalues V) Return Decision Tree.

Begin

Load learning sets first, create decision tree root node 'rootNode', add learning set S into root node as its subset.

For rootNode, we compute Entropy(rootNode.subset) first

If Entropy(rootNode.subset)==0, then rootNode.subset consists of records all with the same value for the categorical attribute, return a leaf node with **decision attribute:attribute value;**

If Entropy(rootNode.subset)!=0, then compute information gain for each attribute left(have not been used in splitting), find attribute A with

Maximum(Gain(S,A)). Create child nodes of this rootNode and add to rootNode in the decision tree.

For each child of the rootNode, apply ID3(S,A,V) recursively until reach node that has entropy=0 or reach leaf node.

End ID3.

3.2 An Example here:

To describe the operation of ID3, we use a classic 'PlayTennis' example.

The symbolic attribute description:

Attribute	**Possible Values**
Outlook	sunny, overcast, rain
Temperature	hot, mild, cool
Humidity	high, normal
Windy	true, false
Decision	n(negative), p(positive)

The Learning set for play tennis example:

Outlook	**Temperature**	**Humidity**	**Windy**	**Decision**
sunny	hot	high	false	n
sunny	hot	high	true	n
overcast	hot	high	false	p
rain	mild	high	false	p
rain	cool	normal	false	p
rain	cool	normal	true	n
overcast	cool	normal	true	p
sunny	mild	high	false	n
sunny	cool	normal	false	p
rain	mild	normal	false	p
sunny	mild	normal	true	p
overcast	mild	high	true	p
overcast	hot	normal	false	p

rain	mild	high	true	n

What we need to do with ID3:

1. Create rootNode, containing the whole learning set as its subset, then computer **Entropy(rootNode.subset) = -(9/14)log$_2$(9/14) -(5/14)log$_2$(5/14)=0.940**

2. Compute information gain for each attribute:

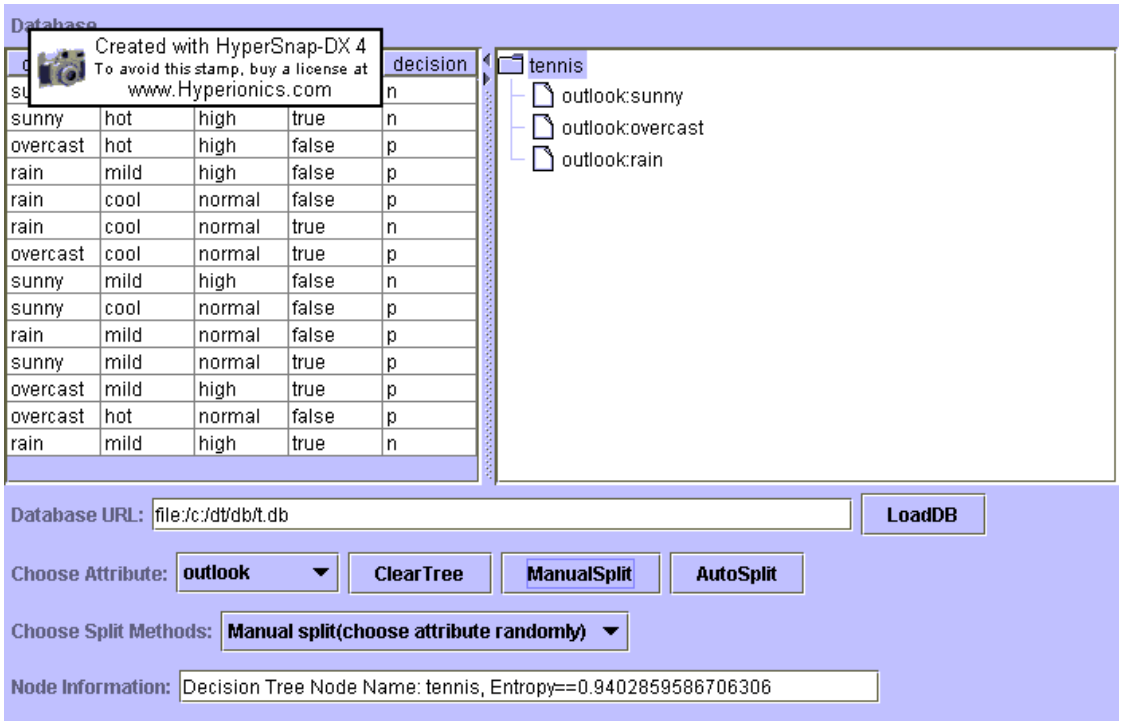
Gain(S,Windy) = Entropy(S)-(8/14)Entropy(S$_{false}$) -(6/14)Entropy(S$_{true}$) =0.048

Gain(S,Humidity) =0.151

Gain(S,Temperature)=0.029

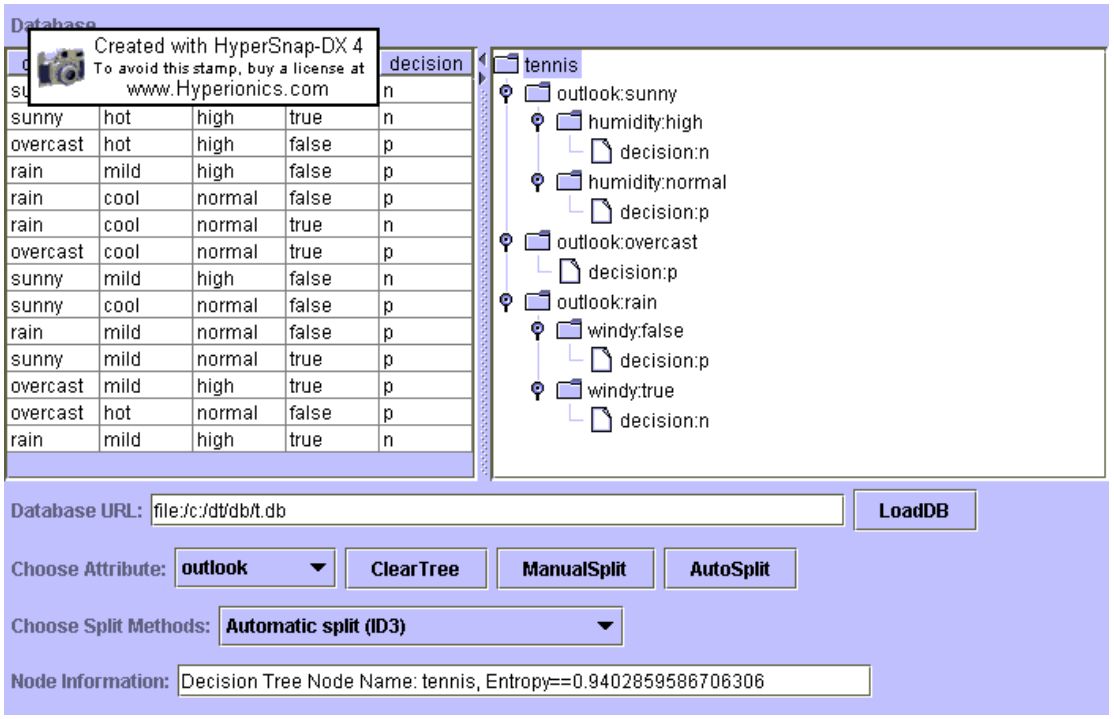
Gain(S, Outlook) = **0.246**

3. Select attribute with the maximum information gain, which is 'outlook' for splitting. We have:(**Note**: the database URL here should be 'http://www.cse.unsw.edu.au/~cs9417ml/DT1/tennis.db' when you use our applet online.)



sunny	hot	high	true	n
overcast	hot	high	false	p
rain	mild	high	false	p
rain	cool	normal	false	p
rain	cool	normal	true	n
overcast	cool	normal	true	p
sunny	mild	high	false	n
sunny	cool	normal	false	p
rain	mild	normal	false	p
sunny	mild	normal	true	p
overcast	mild	high	true	p
overcast	hot	normal	false	p
rain	mild	high	true	n

4.Apply ID3 to each child node of this root, until leaf node or node that has entropy=0 are reached. We have decision tree described as below:



3.3 Dealing with continuous-valued attributes:

Initial definition of ID3 is restricted in dealing with discrete sets of values. It handles symbolic attribute effectively. However, we have to extend it sphere to continuous-valued attributes (numeric attribute) to fit the real world scenario. The algorithm here we use is described at Tom M. Mitchell's 'Machine Learning'. (Tom M. Mitchell,1997,p72-73)

What we have done is to define new discrete valued attributes that partition the continuous-valued attribute into symbolic attribute again.

As an example, we included another database file (tb.db) in which we added numeric attributes. The format is as below:

Outlook	Temperature	Humidity	Windy	Decision
sunny	hot	0.9	false	n
sunny	hot	0.87	true	n
overcast	hot	0.93	false	p
rain	mild	0.89	false	p
rain	cool	0.80	false	p
rain	cool	0.59	true	n
overcast	cool	0.77	true	p
sunny	mild	0.91	false	n
sunny	cool	0.68	false	p
rain	mild	0.84	false	p
sunny	mild	0.72	true	p
overcast	mild	0.49	true	p
overcast	hot	0.74	false	p
rain	mild	0.86	true	n

For humidity attribute, we need to create a new boolean value that is true when humidity<=c and false otherwise. The only thing left is to compute the best threshold c.

In our example, the most information gain is attribute 'outlook', (see 5.2). In the subset rooted at 'outlook:sunny', we would like to compute the information gain for 'Humidity' which is a numeric attribute. To do this, first, we sort humidity as below:

Humidity	0.68	0.72	0.87	0.9	0.91
Playtennis	p	p	n	n	n

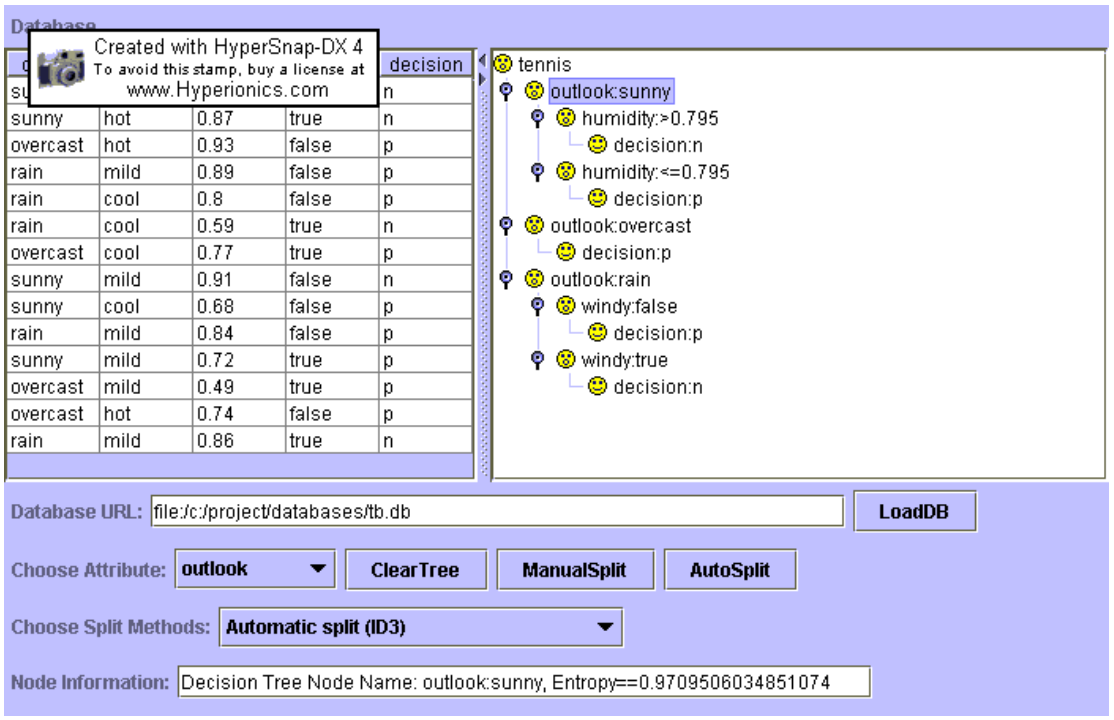
We would like to pick a threshold that produces the greatest information gain. By sorting the numeric attribute values, then identifying adjacent examples that differ in their target classification, we can generate a set of candidate threshold. Then we compute information gain for each candidate and find the best one for splitting. The candidate thresholds for this example is fortunate 1:

Humidity>(0.72+0.87)/2 that is Humidity>0.795, with the information gain G(S,Humidity)=**0.97**. If we

have more than 1 candidate here, we just need to find the best information gain one.

After we include continuous-valued attribute in, the databases and the result of decision tree look like as below:

(**Note**: the database URL here should be 'http://www.cse.unsw.edu.au/~cs9417ml/DT1/tb.db ' when you use our applet online.)



In the end, we also plan to include complex databases such as, 'iris.db', 'german.db', 'zoo.db', etc. These databases can be found in our DT1 directory.

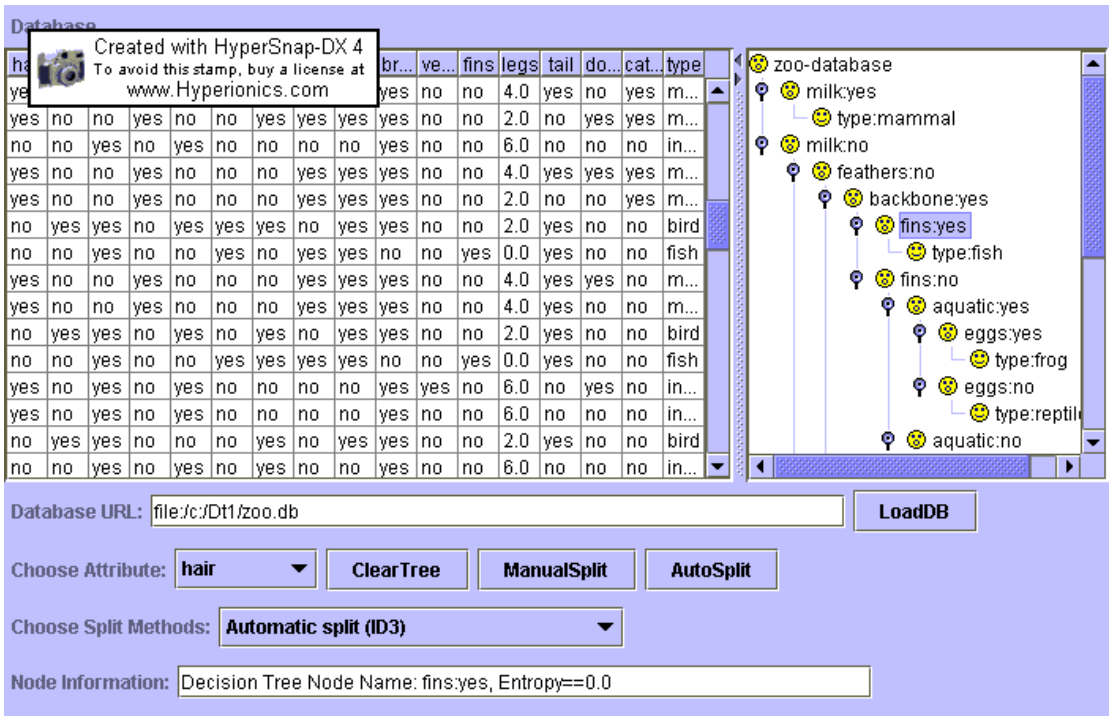
3.4 Extending ID3 to real-valued data:

Just as we mentioned before, ID3 is quite efficient in dealing with the target function that has discrete output values. It can easily deal with instance which is assigned to a boolean decision, such as 'true' and 'false', 'p (positive)' and 'n (negative)'. It is possible to extend target to real-valued outputs.

When we compute information gain in our 'Function.java', what we concern is not just the boolean decision gained from decision tree inductive learning, we try to extend target function to real-valued outputs. Such as in our example,

'zoo.db', the final decision is not simple boolean value but an array of discrete values. You will see the result as below:

(**Note:** The database URL here should be 'http://www.cse.unsw.edu.au/~cs9417ml/DT1/zoo.db' rather than the one displayed as below when you use our applet online)



4 Shortcoming of ID3

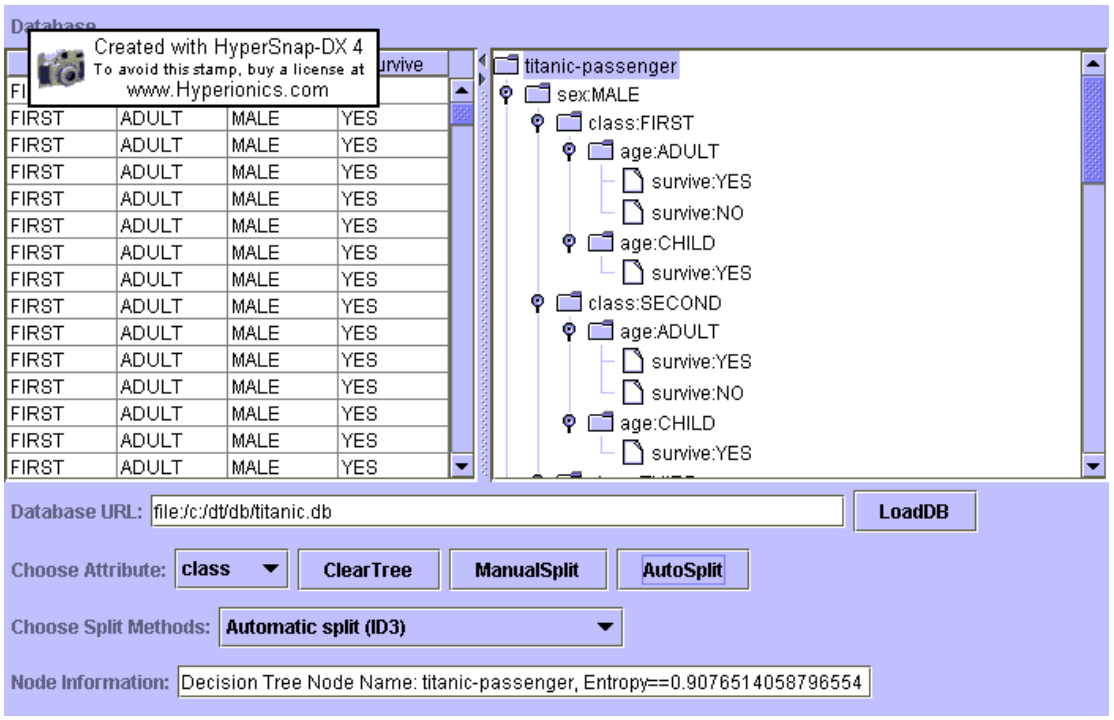
4. 1 Scenario 1:

'A significant shortcoming of ID3 is that the space of legal splits at a node is impoverished. A split is a partition of the instance space that results from placing a test at a decision tree node. ID3 and its descendants only allow testing a single attribute and braching on the outcome of that test' (Paul Utgoff & Carla Brodley, 1990)

During our implementation of ID3, we found that sometimes, we can not gain a result using split creteria provided by ID3. Such as in dealing with 'titanic.db', you may find that we can ot gain a

conclusion at the end. This is described as below(the survive may be 'yes' or 'no' at the end of splitting)

(**Note:** The database URL here should be 'http://www.cse.unsw.edu.au/~cs9417ml/DT1/titanic.db' when you use our applet online)



However, we recommend you to have a look at a decision learning algorithm called PT2, which is an incremental method for finding multivariate splits for decision tree. (Paul Utgoff & Carla Brodley, 1990)

4. 2 Scenario 2:

Another shortcoming is that ID3 relys much on the quality of training data sets. Managing noise of the training sets is of much importance when we need to apply decision tree learning algorithm to the real world. For an example, when there is noise in learning data sets, or when the number of training examples is too small to produce a representative sample of the true target function, ID3 can lead to inaccurate decision making.

A large variety of extensions to the basic ID3 algorithm has been developed in order to apply decision tree learning rules to real world scenario, such as post-pruning trees, handling real-valued

attributes, dealing with missing attributes, using attribute selection standards other than information gain, etc. What we have done here is to implement ID3 and demonstrate different splitting methods, as well as extend ID3 to continuous-valued attributes and apply it to real-valued attribute. We did not cover the issues of pruning which is the task of the Decision Tree Groups 2.

5 Database Format: The database format requirement that can be loaded by our project.

5.1 Database format requirements

We plan to have a simple database format, because the main task of this project is Decision Tree learning algorithm.

Each database is actually a text file. The first line is the name of the database. The second line contains the attribute names immediately followed by their types: numerical or symbolic. A numerical attribute can be real or integer. A symbolic attribute denotes any discrete attribute. Its value can be symbols, numbers or both of them. Each line corresponds to one example and contains the values of the attributes. And During the test stage of our project, we plan to have little numbers of items. However, the database can be expanded dynamically as you wish.

The required database format is addressed as below:

tennis

outlook symbolic temperature symbolic humidity symbolic windy symbolic decision symbolic

sunny hot high false n

sunny hot high true n

overcast hot high false p

rain mild high false p

rain cool normal false p

rain cool normal true n

overcast cool normal true p

sunny mild high false n

sunny cool normal false p

rain mild normal false p

sunny mild normal true p

overcast mild high true p

overcast hot normal false p

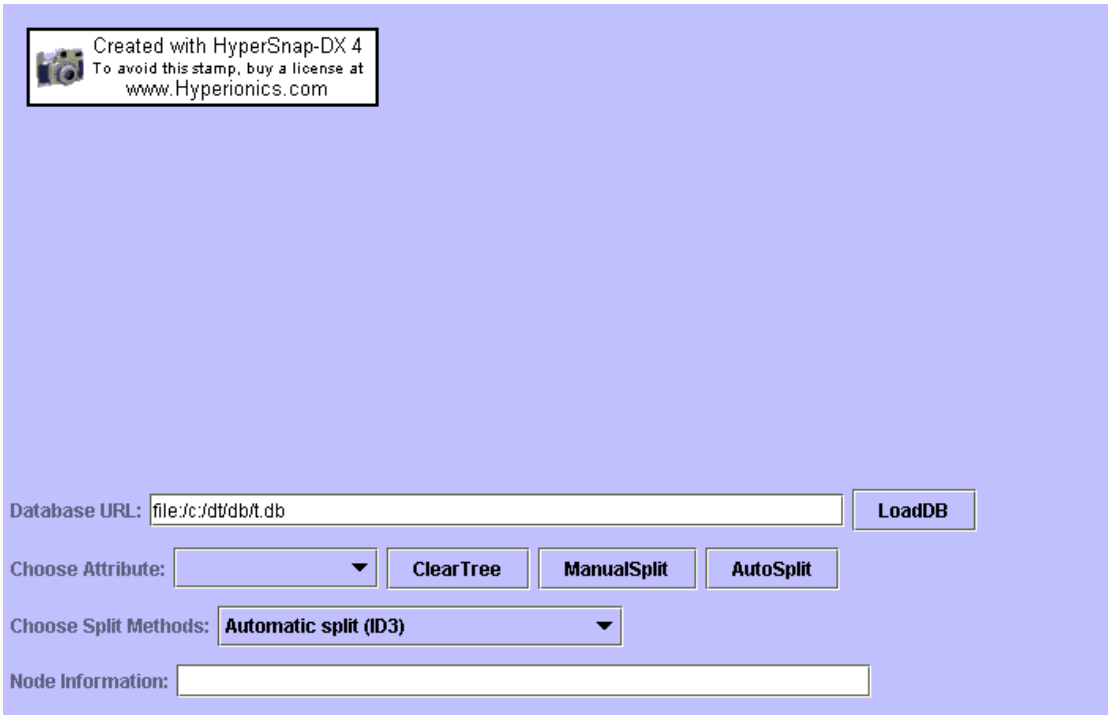
rain mild high true n

5.2 Method of loading database

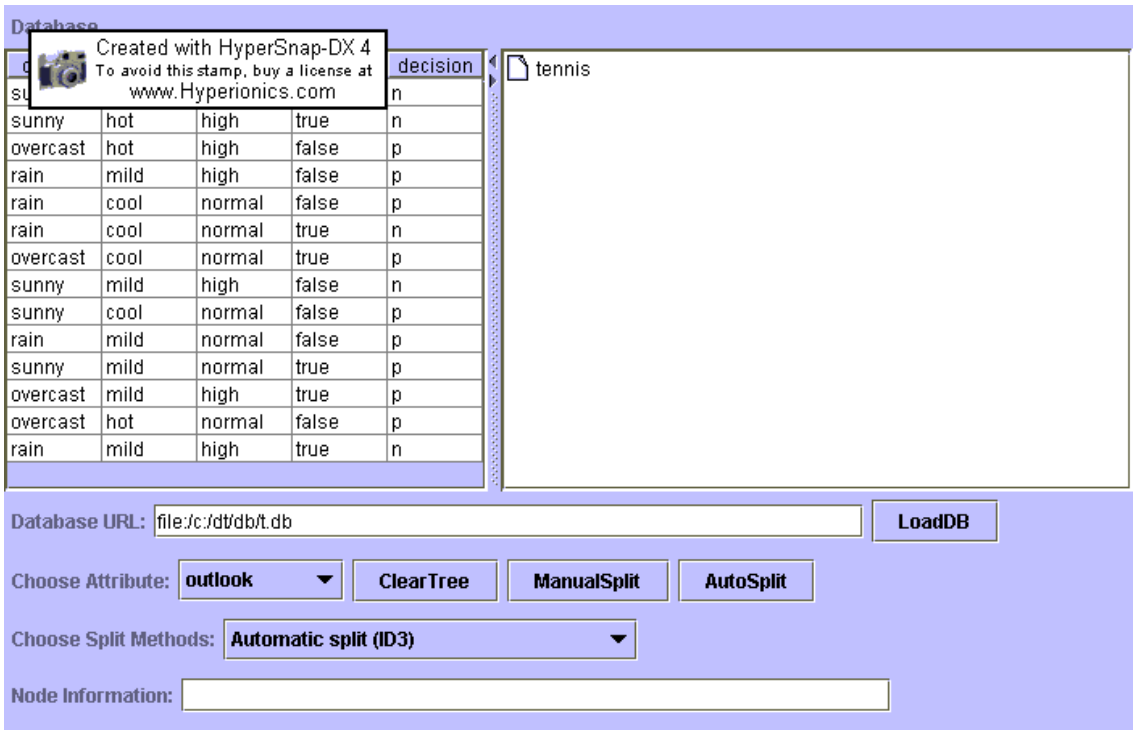
In order to deal with dynamic loading, we use InputStream loading databases file which is actually various learning sets from URL connection.

All we need to do is to input URL into textfield in the applet, and click 'LoadDB' button, we then load the corresponding learning sets in. In the mean time, we can view it on a table in our applet.

Before loading Database ,our applet looks like:



After Loading databases, you can view database table on the left side of splitpane.



5.3 Database that we use

We currently use several databases in this assignment project, one is 'tennis' database, the others are tb.db, titanic.db,iris.db, zoo.db. Each manifests different perspective of our project.

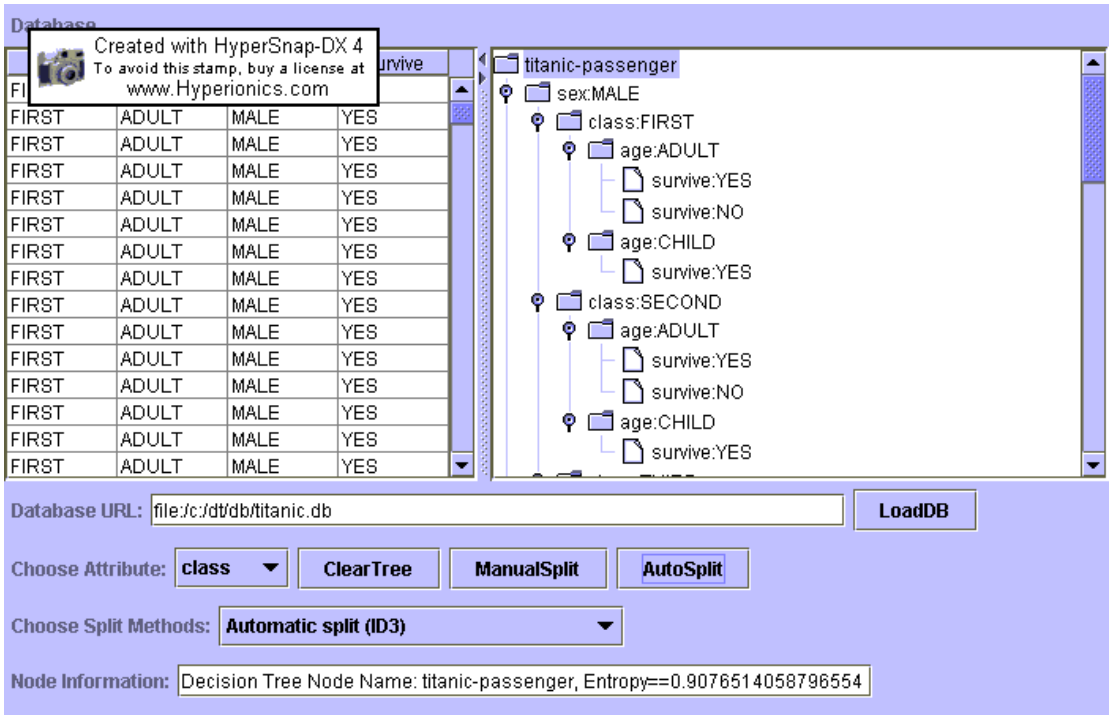
1. tennis.db: display classic ID3, deal with symbolic attribute only.
2. tb.db: extend ID3 to continuous-valued attributes.
3. titanic.db: display the shortcoming of ID3 for its impoverished range of selection of splitting attributes.
4. iris.db: display how our extension to pure numeric attributes.
5. zoo.db: display our capabilities in handling real-valued target ouputs.

After You input database URL ('http://www.cse.unsw.edu.au/~cs9417ml/DT1/tennis.db') in our applet and press 'LoadDB', you can load different database data sets into applet.

As for 'tennis' database, we have discussed a lot in the former part. Here we give a description about 'titanic-passenger' learning sets. The set looks like as below:

Passenger	Class	Age	Sex	Survive
p1	first	adult	male	yes
p2	first	adult	male	yes
:	:	:	:	:
p176	first	adult	female	yes
:	:	:	:	:
p493	second	adult	male	no
:	:	:	:	:
p2201	crew	adult	female	no

The result from ID3 is demonstrated as below:



6 ID3 Applet Demo

6.1 ID3 applet functionality

The general functionalities of this applet are:

1. Dynamic database (learning sets) loading: By inputing URL into 'Database URL' textfield and pushing 'LoadDB' button.

2. Various splitting criteria:

- Automatic split according to ID3 algorithm: After click the AutoSplit button, this will build a decision tree automatically according to ID3.
- Manual split according to ID3: In this mode, you can select node using mouse and push 'ManualSplit' button to split the selected node according to ID3. This mode is actually a step-by-step visualization of automatic split.
- Manual split (randomly choosing attribute): In this mode, we can click the node with mouse, select the intended split method and attribute, then push 'ManualSplit' button.

3. 'ClearTree' button: When error occurs, just click this button to restart demo.

4. Node information display: When you click tree node with mouse, the node name and the entropy of its subset will be displayed at 'Node information' textfield.

5. Database (learning set) information display: In the left hand side of splitpane, there is a table for you to have a view of database information.

6.General instruction displayed at the bottom of applet or your browser.

6.2 ID3 applet demo steps

1. Make sure that your browser support Java 2 (Javax.swing.*)--- at least netscape6, or else you can run 'appletviewer.exe'. **A good news is we can use a browser called 'Konquerer' in UNSW cse labs, the only thing we need is a little bit patience because it is really slow in loading applet. Make sure to close 'security manager' option in browser configuration.:)**
2. When you can run appletviewer, make sure that 9417.html and all class file in the same directory and use command line: appletviewer 9417.html
3. Load database first: Input correct URL (http://www.cse.unsw.edu.au/~cs9417ml/DT1/tb.db) of our training database in textfield named 'Database URL' and push 'LoadDB' button, then you will notice that you can view database and attribute information in applet. We have several databases available :

'tennis.db', 'tb,db', 'titanic.db', 'iris.db', 'zoo.db'. Each displays different perspective of our project.

4. Automatic build decision tree: Select 'Automatic split(ID3)' from JComBobox 'Choose Split Methods', then click 'AutoSplit' button, the returned decision tree will be displayed at right scrollpane of splitpane.
5. Manual build decision tree using ID3:
    - Before start manual split, click 'ClearTree' button to clear the decision tree built from the following step;
    - Select tree node with mouse;
    - Select split methods 'Manual split(ID3)';
    - Click 'ManualSplit' button;
    - Select intended node, then click 'ManualSplit' button until you reach leaf node.
6. Manual build decision tree with randomly selected attribute::
    - Before start manual split, click 'ClearTree' button to clear the decision tree built from the following step;
    - Select tree node with mouse;
    - Select split methods 'Manual split(ID3)';
    - Choose attribute in JComBobox 'Choose attribute' for splitting;
    - Click 'ManualSplit' button;
    - Select intended node and the right attribute, then click 'ManualSplit' button until you reach leaf node;
    - Whenever you select tree node with you mouse, you can view node information at 'Node information' textfield.

6.3 Applet Demo

ID3 Applet Demo (here is the hyperlink of the java applet simulation of decision tree ID3. However this link unavailable this moment, please contact me at weipengtiger@hotmail.com for the source codes)

Acknowledgements:

We wish to thank for Professor Claude Sammut and Associate professor Achim Hoffmann at UNSW for their guidance and support for our project in the course of Machine Learning. We also appreciate the other project groups that

involved in providing helpful evaluations and suggestions. Without our friend Tony Li's help, this report would have been still in HTML and Word files. We would thank him in assisting me to convert this report to PDF version.

Reference

1. Tom M. Mitchell, (1997). ***Machine Learning,*** **Singapore, McGraw-Hill.**

2. Paul E. Utgoff and Carla E. Brodley, (1990). 'An Incremental Method for Finding Multivariate Splits for Decision Trees', ***Machine Learning: Proceedings of the Seventh International Conference,*** **(pp.58). Palo Alto, CA: Morgan Kaufmann**