Université de Tunis

Institut Supérieur de Gestion de Tunis

MASTÈRE EN INFORMATIQUE APPLIQUÉE A LA GESTION

Pruning Belief Decision Trees

Elaboré par:
Salsabil Trabelsi

Supervisé par:
Pr. Khaled Mellouli (IHEC, Tunis)
M. Zied Elouedi (ISG, Tunis)

Juin 2006

Acknowledgements

I would like to thank my advisor Professor Khaled Mellouli for introducing me to an interesting area of research. I thank him for his continuous guidance and for all the valuable discussions we have had during the period of work.

I would like to express my sincere gratitude to my co-advisor Zied Elouedi in the fulfillment of this master thesis. He has been very patient in supervising this work at all its stages. His competence and his generosity have allowed me to work in ideal conditions. His comments and his observations have always been of great interest. I thank him for his support and for his constant encouragements.

I am also thankful to Thierry Denoeux from *Université de Technologie de Compiègne* for replying me each time I asked him questions regarding belief function theory.

Finally, my thanks go to all member of my laboratory LARODEC in *Institut Supérieur de Gestion de Tunis* and to all my friends: Abir, Sarra, Bakhta, Hanen, wided..., to my family for their moral support.

Contents

List of Tables

List of Figures

Introduction

Databases are rich with hidden information that can be used for making intelligent business decisions. Classification is a form of data analysis that can be used to predict values of a categorical dependent variable (class, group membership, etc.) from one or more predictor variables.

There is a number of techniques for analyzing classification problems which are widely used in artificial intelligence like decision trees, artificial neural networks, K-nearest neighbors, naive Bayesian networks, etc. These techniques are applied in several real-world applications such as marketing, medical diagnosis, credit approval, etc.

As mentioned earlier, there is a large number of techniques that an analyst can choose from when analyzing classification problems. Decision trees, when they "work" and produce accurate predictions or predicted classifications based on few logical if-then conditions, have a number of advantages over many of those alternative techniques.

In most cases, the interpretation of results summarized in a tree is very simple. This simplicity is not only useful for purposes of rapid classification of new observations, but can also often yield a much simpler "model" for explaining why observations are classified or predicted in a particular manner.

However, this standard decision tree like the other classification techniques do not well perform their classification task in an environment characterized by uncertainty and imprecision due to noise or residual variation in data. Hence, having uncertain or imprecise information relative to any parameters of classification problem may lead to poor classification results and even to the impossibility to ensure the classification task.

In order to overcome this limitation, many researches have been done to adapt standard decision tree to this kind of environment. The idea was to introduce theories that could represent uncertainty. Several kinds of decision trees were developed: **probabilistic decision trees** (Quinlan, 1987, 1990), **fuzzy decision trees** (Umano et al., 1994), (Zeidler & Schlosser, 1996), (Marsala, 1998), **belief decision trees** (Elouedi et al., 2000)(Elouedi et al., 2001), (Denoeux & Skarstien-Bjanger, 2000) and **possibilistic decision trees** (Hüllermeier, 2002), (Ben Amor et al., 2004), (Jenhani et al., 2005) .

The belief decision tree approach is a decision tree technique adapted in order to handle uncertainty about the actual class of the objects in the training set and also to classify objects characterized by uncertain attributes. The uncertainty is represented by the Transferable Belief Model (TBM) one interpretation of the belief function theory. In our work, we will focus in belief decision trees.

When a decision tree is built, many of branches will reflect anomalies in the training data due to noise and outliers. Tree pruning methods address this problem of overfitting or overlearning the data. Such methods typically use statistical measure to remove the least reliable branches, generally resulting in faster classification and an improvement in classification accuracy.

Two kinds of pruning can be applied:

1. The pre-pruning: Stop growing the tree before it perfectly classifies the training set.

2. The post-pruning: Grow the full tree, allow it overfit the data and then post-prune it.

A post-pruning approach requires more computation than pre-pruning, yet generally leads to a more reliable tree.

So, the beneficial effects of a post-pruning approach have attracted the attention of many researches, who proposed a number of methods including **minimal cost-complexity pruning** (Breiman et al., 1984), **reduced error pruning** (Quinlan, 1987), **critical value pruning** (Mingers, 1987),

pessimistic error pruning (Quinlan, 1987), **minimum error pruning** (Niblett & Bratko, 1986) and **error based pruning** (Quinlan, 1993).

All these methods deal with standard decision trees and not with the other variants.

When a belief decision tree (BDT) is built, many of branches will reflect anomalies in the training data due to noise and outliers. Our aim is to overcome this problem of overfitting in belief decision trees. In order to reduce the size of the decision tree and to improve the classification accuracy.

Our Objective concerns pruning belief decision trees : we propose to develop a pruning belief decision tree method. Pruning can improve the comprehensibility of a belief decision tree and its accuracy on unseen cases.

Our master thesis focuses on post-pruning methods in belief decision tree. So, we suggest to develop post-pruning methods to simplify the belief decision trees in order to reduce the size and the complexity and also to increase the quality of classification.

This report is organized in four chapters belonging to two main parts:

Part I: *Theoretical aspects* presents the necessary theoretical aspects concerning the belief function theory, decision trees, pruning methods and belief decision trees which are detailed in chapter 1 and chapter 2.

Part II: *Pruning Belief Decision Trees* details the pruning of belief decision trees. Chapter 3, presents two pruning belief decision tree methods within averaging and conjunctive approaches, that we have developed. Chapter 4 is related to the implementation and simulations which have been performed in order to analyze and evaluate results given by our proposed pruning methods.

Finally, a conclusion summarizes all the works presented in this report and proposes further works that may be done to improve our methods.

One appendix is provided at the end of this report. It presents the different variables and algorithms of our pruning methods in both approaches.

Part I

Theoretical Aspects

Chapter 1

Belief function theory

1.1 Introduction

The theory of belief functions is considered as a useful theory for representing and managing uncertain knowledge because of its relative flexibility. This theory is introduced by Dempster (1968) and Shafer (1976) as a model to represent beliefs where it is usually called the Dempster-Shafer theory.

The belief function theory is widely applied in artificial intelligence. There exist several interpretations of this theory: in particular **the lower probability model** (Walley, 1991), **the Dempster's model** (Dempster, 1967, 1968), **the theory of hints** (Kholas & Monney, 1995), and **the Transferable Belief Model (TBM)**(Smets, 1988, 1998a; Smets & Kennes, 1994; Smets & Kruse, 1997).

This theory is appropriate to handle uncertainty in classification problems especially within the decision tree technique. Belief decision trees deal with the interpretation of the belief function theory as explained by the TBM.

This chapter presents an overview of some of the concepts of the belief function theory. We have based our work on the TBM. Consequently, the presentation in this chapter will be based on this model. Several concepts are detailed like combination and discounting. The different notations are illustrated by examples.

1.2 Basic concepts

1.2.1 Frame of discernment

Let Θ be a finite set of elementary events to a given problem, called **the frame of discernment**. It also referred to as **the universe of discourse** or **the domain of reference** (Smets, 1988). All the subsets of Θ belong to the power set of Θ, denoted by 2^{Θ}.

Example 1.1 *Let's treat the problem of identification of the murderer. Suppose the frame of discernment related to this problem is defined as follows:* $\Theta = \{Pierre, Paul, Marie\}$

The power set of Θ *is:*
$2^{\Theta} = \{\emptyset, \{Pierre\}, \{Paul\}, \{Marie\}, \{Pierre, Paul\}, \{Paul, Marie\}, \{Pierre, Marie\}, \Theta\}$

1.2.2 Basic belief assignment

The impact of a piece of evidence on the different subsets of the frame of discernment Θ is represented by **basic belief assignment** (bba).

The bba is defined as follows:

$$\mathrm{m}:2^{\Theta} \rightarrow [0,1]$$

$$\sum_{A \subseteq \Theta} m(A) = 1 \tag{1.1}$$

The value *m(A)*, named a **basic belief mass** (bbm), represents the portion of belief committed exactly to the event A.

Example 1.2 *Assume* $\Theta = \{Pierre, Paul, Marie\}$
The bba related to a piece of evidence concerning the murderer is defined as follows:

$m(\{Pierre\})=0.2;$
$m(\{Pierre, Paul\})=0.5;$
$m(\Theta)=0.3;$

For example, 0.2 represents the part of belief exactly supporting that the murderer is Pierre.

1.2.3 Belief function

A belief function, denoted *bel*, corresponding to a specific bba m, assigns to every subset A of Θ the sum of masses of belief committed to every subset of A by m (Shafer, 1976).

The belief function *bel* is defined as follows:

$$bel(A) = \sum_{\emptyset \neq B \subseteq \Theta} m(B) \tag{1.2}$$

Example 1.3 *The belief function bel corresponding to the bba m (see example 1.2) is defined as follows:*

bel(∅)=0;
bel({Pierre})=0.2;
bel({Paul})= bel({Marie})=bel({Paul, Marie})=0;
bel({Pierre, Paul})=0.2+0.5=0.7;
bel({Pierre, Marie})=0.2;
bel(Θ)=0.2+0.5+0.3=1;

For example, 0.7 is the total belief committed to the proposition {Pierre, Paul}.

1.2.4 Plausibility function

The plausibility function *pl* quantifies the maximum amount of belief that could be given to a subset A of the frame of discernment. It is equal to the sum of the bbm's relative to subsets B compatible with A.

The plausibility function *pl* is defined as follows:

$$pl(A) = \sum_{A \cap B \neq \emptyset} m(B) \tag{1.3}$$

Example 1.4 *The plausibility function pl corresponding to the bba m (see example 1.2) is defined as follows:*

pl(∅)=0;
pl({Pierre})=0.2+0.5+0.3=1;
pl({Paul})=0.5+0.3=0.8;
pl({Marie})=0.3;
pl({Pierre,Paul})=0.2+0.5+0+0.3=1;
pl({Pierre, Marie})=0.5+0.3=1;
pl({Paul, Marie})=0.2+0.5+0.3=1;
pl(Θ)=0.2+0.5+0.3;

For example, 0.3 represents the maximum degree of belief that the proposition {Marie} may have.

1.2.5 Commonality function

Another function is used basically to simplify computations namely **the commonality function *q*** (Branett, 1991) is defined as follows:

$$q(A) = \sum_{A \subseteq B} m(B) \tag{1.4}$$

Examlpe 1.5 *The commonality function q corresponding to the bba m (see example 1.2) is defined as follows:*

q(∅)=1;
q({Pierre})=0.2+0.5+0.3=1;
q({Paul})=0.5+0.3=0.8;
q({Marie})=0.3;
q({Pierre, Paul})=0.5+0.3=0.8;
q({Pierre, Marie})=q({Paul, Marie})=0.3;
q(Θ)=0.3;

Remark:

The basic belief assignment (m), the belief function (bel), the plausibility function (pl) and the commonality function (q) are considered as different expressions of the same information (Denoeux, 1999).

1.2.6 Focal elements, body of evidence, core

The subsets A of the frame of discernment Θ such that that *m(A)* is strictly positive are **the focal elements** of the bba m.

The pair (F,m) is called **a body of evidence** where F is the set of all the focal elements relative to the bba m.

The union of all the focal elements of m are named **the core** and are defined as follows:

$$\varphi = \bigcup_{A:m(A)>0} A \tag{1.5}$$

Example 1.6 *Let's continue with the example 1.2, the subsets* $\{Pierre\}$, $\{Pierre, Paul\}$, *and* Θ *are the focal elements of the bba m.*

So, (F,m) is called the body of evidence such that: $F=\{\{Pierre\}, \{Pierre, Paul\}, \Theta\}$

The core of this m is defined as follows:
$\varphi = \{Pierre\} \cup \{Pierre, Paul\} \cup \Theta = \Theta$

1.3 Special belief functions

In the literature, several kinds of belief functions are proposed. Such functions are used to express particular situations related generally to uncertainty.

1.3.1 Vacuous belief function

A vacuous belief function is a normalized belief function defined such that (Shafer, 1976):

$$m(\Theta) = 1 \; and \; m(A) = 0 \; for \; A \neq \Theta \tag{1.6}$$

Such basic belief assignment, quantifies the state of **total ignorance**.

Example 1.7 *Assume an expert was not able to detect the murderer. Hence, we get a state of total ignorance where the corresponding bba is defined as follows:*

$m(\Theta)=1$ *and* $m(A)=0$ *for* $A \neq \Theta$

1.3.2 Categorical belief function

A categorical belief function is a normalized belief function such that its bba is defined as follows (Mellouli, 1987):

$$m(A) = 1 \, for \, some \, A \subset \Theta \, and \, m(B) = 0, \; for B \subseteq \Theta, B \neq A \tag{1.7}$$

Example 1.8 *Assume an expert was certain that the murderer is a man. So, the corresponding bba presents a categorical belief function defined as follows:*

m({Pierre, Paul})=1;

1.3.3 Certain belief function

A certain belief function is a categorical belief function such that its focal element is a singleton. Its corresponding bba is defined as follows:

$$m(A) = 1 \; and \; m(B) = 0 \, for \, all \, B \neq A \, and \; B \subseteq \Theta \tag{1.8}$$

Such function represents a state of total certainty as it assigns all the belief to a unique elementary event.

Example 1.9 *Assume an expert affirms that the murderer is Marie. So, the corresponding bba presents a certain belief function defined as follows:*

m({Marie})=1;

1.3.4 Bayesian belief function

A belief function is said to be **Bayesian** if its focal elements are singletons (Shafer, 1976). Hence, *bel* becomes a probability distribution. It is defined as follows:

$$bel(\emptyset) = 0 \tag{1.9}$$

$$bel(\Theta) = 1 \tag{1.10}$$

$$bel(A \cup B) = bel(A) + bel(B) \; Where \; A, \; B \; \subset \Theta \, and \, A \cap B = \emptyset \tag{1.11}$$

Example 1.10 *Let's consider* Θ=*{Pierre, Paul, Marie}.*
We get a piece of evidence expressed by the following bba m:

m({Pierre})=0.4;
m({Paul})=0.5;
m({Marie})=0.1;
m(Θ)=0;

The bba m is a Bayesian bba since all its focal elements are singletons.

1.3.5 Consonant belief function

A belief function is said to be **consonant** if its focal elements (A_1, A_2, ...,A_n) are nested, that is $A_1 \subseteq A_2 \subseteq ..... \subseteq A_n$

Example 1.11 *Let's consider the same bba defined in the example 1.2:*

m({Pierre})=0.2;
m({Pierre, Paul})=0.5;
m(Θ)=0.3;

The focal elements of this bba m are nested. Hence, it is a consonant bba.

1.3.6 Dogmatic and non-dogmatic belief functions

A belief function is said to be **dogmatic** if and only if its corresponding bba m such that $m(\Theta) = 0$. This case involves some previous cases (certain belief functions, categorical belief functions). **A non-dogmatic belief function** is defined such that $m(\Theta) \succ 0$ (Smets, 1995).

1.4 Rules of combination

In the transferable belief model, the basic belief assignments induced from distinct pieces of evidence can be combined by the conjunctive rule or disjunctive rule.

1. **The conjunctive rule :** When we know that both sources of information are fully reliable then the bba representing the combined evidence satisfies (Smets, 1998b):

$$(m_1 \textcircled{\cap} m_2)(A) = \sum_{B,C \subseteq \Theta : B \cap C = A} m_1(B) m_2(C) \qquad (1.12)$$

This rule can be simply computed in terms of the commonality functions as follows:

$$(q_1 \textcircled{\cap} q_2)(A) = q_1(A)q_2(A) \tag{1.13}$$

where q_1 and q_2 are respectively the commonality functions corresponding respectively to the bba's m_1 and m_2.

The conjunctive rule is considered as unnormalized Demspter's rule of combination dealing with the closed world assumptions, defined as follows (Shafer, 1976, 1986)

$$(m_1 \oplus m_2)(A) = K(m_1 \textcircled{\cap} m_2)(A) \tag{1.14}$$

Where

$$K^{-1} = 1 - (m_1 \textcircled{\cap} m_2)(\emptyset) \tag{1.15}$$

$$and\ (m_1 \oplus m_2)(\emptyset) = 0 \tag{1.16}$$

K is called *the normalization factor.*

Properties

The conjunctive rule of combination is characterized by the following properties:

- Compositionality:
  $(m_1 \textcircled{\cap} m_2)$(A) is function of A, m_1 and m_2
- Commutativity:
  $$m_1 \textcircled{\cap} m_2 = m_2 \textcircled{\cap} m_1 \tag{1.17}$$
- Associativity:

$$(m_1 \textcircled{\cap} m_2) \textcircled{\cap} m_3 = m_1 \textcircled{\cap} (m_2 \textcircled{\cap} m_3) \tag{1.18}$$

- Non-idempotency:
  The conjunctive rule of combination is not idempotent. So,

$$(m \textcircled{\cap} m) \neq m \tag{1.19}$$

- Neutral element:
  The neutral element within the conjunctive rule of combination is the vacuous basic belief assignment representing the total ignorance.

$$(m \textcircled{\cap} m_0) = m \tag{1.20}$$

  Where m_0 is a vacuous bba.

2. **The disjunctive rule :** When we only know that at least one of sources of information is reliable but we do not know which is reliable, then the bba representing the combined evidence satisfies (Smets, 1998b):

$$(m_1 \textcircled{\cup} m_2)(A) = \sum_{B,C \subseteq \Theta : B \cup C = A} m_1(B) m_2(C) \tag{1.21}$$

Example 1.12 *Assume* Θ=$\{$*Pierre, Paul, Marie*$\}$
Let two bba's m_1 *and* m_2 *relative to two pieces of evidence.*
m_1*(*$\{$*Pierre*$\}$*)=0.6;*
m_1*(*$\{$*Pierre,Paul*$\}$*)=0.2;*
m_1*=(*Θ*)=0.2;*

m_2*=(*$\{$*Paul*$\}$*)=0.4;*
m_2*=(*$\{$*Pierre, Paul*$\}$*)=0.3;*
m_2*=(*Θ*)=0.3;*

Applying the conjunctive rule of combination, we get:

($m_1 \textcircled{\cap} m_2$)($\emptyset$)=0.24;
($m_1 \textcircled{\cap} m_2$)($\{$*Pierre*$\}$)=0.36;
($m_1 \textcircled{\cap} m_2$)($\{$*Paul*$\}$)=0.16;
($m_1 \textcircled{\cap} m_2$)($\{$*Pierre, Paul*$\}$)=0.18;
($m_1 \textcircled{\cap} m_2$)($\Theta$)=0.06;

Applying the disjunctive rule of combination, we get:

($m_1 \textcircled{\cup} m_2$)($\{$*Pierre, Paul*$\}$)=0.56;
($m_1 \textcircled{\cup} m_2$)($\Theta$)=0.44;

1.5 Discounting

In the transferable belief model, discounting allows to take in consideration the reliability of the information source that generates the bba m. For $\alpha \in [0,1]$, let (1-α) be the degree of confidence ('reliability') we assign to the source of information. If the source is not fully reliable, the bba it generates is 'discounted' into a new less informative bba denoted m^α:

$$m^\alpha(A) = (1-\alpha).m(A), \qquad for\, A \subset \Theta \tag{1.22}$$

$$m^\alpha(\Theta) = \alpha + (1-\alpha).m(\Theta) \tag{1.23}$$

Example 1.13 *Let's discount the bba m (given in the example 1.2) with degree of reliability is equal to 0.9, so we get:*

$m^\alpha(\{Pierre\})$*=0.9*0.2=0.18;*
$m^\alpha(\{Pierre, Paul\})$*=0.9*0.5=0.45;*
$m^\alpha(\Theta)$*=0.1+(0.9*0.3)=0.37*;

1.6 Pignistic probability

In the transferable belief model, holding beliefs and making decisions are distinct processes. Hence, it proposes a two level models:

- *The credal level* where beliefs are entertained and represented by belief functions.
- *The pignistic level* where beliefs are used to make decisions and represented by probability functions called the pigninstic probabilities and is defined as:

$$BetP(A) = \sum_{B \subseteq \Theta} \frac{\mid A \cap B \mid}{\mid B \mid} \frac{m(B)}{(1-m(\emptyset))},\ for\, all\, A \in \Theta \tag{1.24}$$

Example 1.14 *In order to make a decision, we have to compute the pignistic probability BetP corresponding to the bba m (given in the example 1.2), we get:*

BetP({Pierre})=0.55;
BetP({Paul})=0.35;
BetP({Marie})=0.1;

It is more probable that the murderer is Pierre.

1.7 Conclusion

In this chapter, we have presented the basic concepts of belief function theory as understood in the Transferable Belief Model.

This theory is appropriate to handle uncertainty in classification problems especially within the decision tree technique. The following chapter will deal with the technique of decision tree where its basics will be described.

Chapter 2

Decision trees

2.1 Introduction

Decision trees are a simple yet successful technique for supervised classification learning. They have the specific aim of allowing us to predict the class of an object given known values of its attributes.

A decision tree is a flow-chart-like tree structure. The visual presentation makes the decision tree model very easy to understand and assimilate. It has also good classification accuracy compared to other classification techniques.

Decision trees are applied successfully in many areas such as expert systems, medical diagnoses, speech recognition, etc. They can also be used in other fields like marketing, finance, industry, etc.

In this chapter, we are interested to the basics of decision trees. We focus on standard decision trees where their representation, procedures and some algorithms will be described.

In the second part of this chapter, we present the pruning methods.

The third part of this chapter, deals with belief decision trees. This approach is based on both the decision tree technique and belief function theory in order to cope with uncertainty. We present the two major procedures related to the belief decision tree technique.

2.2 Standard decision tree

2.2.1 Notations

A typical problem of classification is described as follows:

- **Object universe (U):** It contains all the objects already classified or to classify.
  U = $\{I_1, I_2, ...\}$
- **Attributes (A):** They describe each instance of object universe U. In fact, each attribute is considered as one property of the object (e.g.: color, age, height, etc.).
  A = $\{A_1, A_2, ..., A_m\}$
- **Classes (C):** They are represented by the set of classes in which each object of the universe U has belong.
  C = $\{C_1, C_2, ..., C_n\}$
- **Training set (T):** It includes all the objects whose classes are known.

The main objective is to find a general classification rule that works well on the objects in a given training set (Quinlan, 1990a) in order to be useful to correctly classify new objects.

2.2.2 Decision tree representation

A decision tree is a flow-chart-like tree structure, which is composed of three basic elements:

1. **A decision node** specifying a test attribute.
2. **An edge** or **a branch** corresponding to the one of the possible attribute values which means one of the test attribute outcomes.
3. **A leaf** which is also named an answer node, including objects that, typically, belong to the same class, or at least are very similar.

Example 2.1 *In a bank, given a client's information, a decision tree could be used to determine the client's category. This classification will be useful since it allows to the bank to plan its loan policy.*

- *The object universe:* $U = \{$*client1, client2,...*$\}$
- $A = \{$*Income, Property, Unpaid-credit*$\}$
  Each attribute may take the following values:

  - Income with possible values $\{$*No, Low, Average, High*$\}$
  - Property with possible values $\{$*Less, Greater*$\}$
  - Unpaid-credit with possible values $\{$*Yes, No*$\}$
- *Three classes are defined by the bank and one of them will be assigned to each client asking for a loan:*

  - C_1*: accept to give the whole loan.*
  - C_2*: accept to give a part of the loan.*
  - C_3*: refuse to give the loan.*
  $C = \{C_1, C_2, C_3\}$*;*

We could have this kind of decision tree (see Figure 2.1):

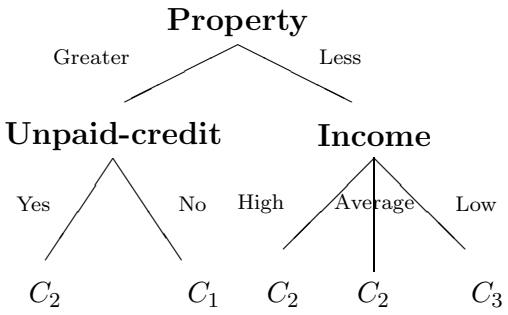

Figure 2.1: Decision tree

2.2.3 Decision tree procedures

Decision trees procedures consist of two steps building a decision tree using a training data set and then using the tree to classify unseen cases.

Building step

Building step goes through two phases: a tree-growing (splitting) phase followed by a pruning phase.

The tree-growing phase is an iterative process, which involves splitting the data into progressively smaller subsets. Each iteration considers the data in only one node.

Exploring the tree model may reveal nodes that are undesirable because of overfitting. Pruning step is used to make a tree more general.

There are two kinds of pruning that can be applied:

1. The pre-pruning: applied while the decision tree is built.
2. The post-pruning: applied after the construction of the tree.

Classification step

Decision trees classify instances by sorting them down the tree from the root node to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of an attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute.

2.2.4 Decision tree algorithms

Several algorithms for building decision trees have been developed. The most popular ones are: simple **ID3** and its successor **C4.5** algorithm developed by Quinlan (1986, 1993). We can also mention the **CART** algorithm of Breiman et al. (1984).

In addition to these non-incremental algorithms, many incremental building decision trees algorithms have been proposed such as **ID4** of Schlimmer and Fisher (1986) and notably **ID5** and **ID5R** of Utgoff (1988, 1989).

A generic decision tree algorithm is characterized by the next parameters:

1. **The attribute selection measure:** A critical parameter, generally based on the information theory (Shannon, 1948), for building decision trees. It allows determining the best attribute for each node, in order to partition the training set T. We can for instance mention those suggested by Quinlan: **the information gain** (Quinlan, 1986) and **the**

gain ratio (Quinlan, 1993). Other measures are also developed namely **the Lopez De Mantaras** measure (Lopes De Mantaras, 1991), **the normalized gain** (Jun & kim, 1997).

2. **The partitioning strategy:** The current training set will be divided by taking into account the selected test attribute.

3. **The stopping criteria:** Generally, we stop the partitioning process if one of the following situations is presented:
   - All the remaining objects belong to only one class.
   - No further attribute to test.
   - All the remaining attributes have no informational contribution.

The choice of these parameters makes the major difference between decision tree algorithms.

The C4.5 algorithm

C4.5 is a supervised learning algorithm developed by Quinlan as successor of ID3 algorithm. The C4.5 algorithm uses **the gain ratio criterion** as the attribute selection measure. This parameter is expressed as follows:

$$\text{Gain ratio(T, A)} = \frac{Gain(T, A)}{SplitInfo(T, A)} \tag{2.1}$$

$$Gain(T, A) = Info(T) - Info_A(T) \tag{2.2}$$

$$\text{Where Info(T)} = -\sum_{i=1}^{n} \frac{freq(C_i, T)}{|T|} \log_2 \frac{freq(C_i, T)}{|T|} \tag{2.3}$$

$$\text{and } Info_A(T) = \sum_{v \in D(A)} \frac{|T_v^A|}{|T|} Info(T_v^A) \tag{2.4}$$

$$\text{Split Info(T, A)} = -\sum_{v \in D(A)} \frac{|T_v^A|}{|T|} \log_2 \frac{|T_v^A|}{|T|} \tag{2.5}$$

Gain (T, A) is defined as the expected reduction in entropy (impurity) resulting from partitioning T according to A.

Split Info(T, A) represents the potential information generated by dividing T into n subsets. So, Gain ratio(T, A) expresses the proportion of information generated by the split that appears helpful for classification.

The gain ratio of each attribute is computed and the one presenting the highest value will be selected.

The different steps of the C4.5 algorithm are summarized as follows:

1. If all instances belong to only one class, then the decision tree is a leaf labeled with that class.
2. Otherwise,
   - Select the attribute that maximizes Gain ratio(T, A).
   - Split the training set T into several subsets, one for each value of the selected attribute.
   - Apply the same procedure recursively for each generated subset.

Example 2.2 *This a simple example in order to explain the C4.5 algorithm.*

Let T be the training set composed by ten objects which are characterized by three attributes:

- *Income* = {*Low, Average, High*}
- *Property* = {*Less, Greater*}
- *Unpaid-credit* = {*Yes, No*}

Three classes are possible $\{C_1, C_2, C_3\}$. *T is described as follows (see Table 2.1):*

Table 2.1: Training set

Income	Property	Unpaid-credit	Class
High	Greater	No	C_1
High	Greater	Yes	C_2
High	Greater	No	C_1
High	Less	Yes	C_2
Average	Greater	No	C_3
Average	Greater	Yes	C_2
Average	Less	No	C_2
Average	Less	Yes	C_2
Low	Less	No	C_3
Low	Less	Yes	C_3

Let's compute the value of Info(T) relative to the training set T.

Info(T)=$-\frac{freq(C_1,T)}{|T|}\log_2\frac{freq(C_1,T)}{|T|}-\frac{freq(C_2,T)}{|T|}\log_2\frac{freq(C_2,T)}{|T|}-\frac{freq(C_3,T)}{|T|}\log_2\frac{freq(C_3,T)}{|T|}$
=1.485;

In order to compute the information gain of each attribute, let us, first, compute $Info_{Income}(T)$, $Info_{Property}(T)$ *and* $Info_{Unpaid-credit}(T)$.

$$Info_{Income}(T) = \frac{4}{10}Info(T^{Income}_{High})+\frac{4}{10}Info(T^{Income}_{High})+\frac{2}{10}Info(T^{Income}_{High}) = 0.725;$$

Gain(T, Income)=Info(T)-$Info_{Income}(T)$*=0.761;*

Split Info(T, Income)=$-\frac{4}{10}\log_2\frac{4}{10}-\frac{4}{10}\log_2\frac{4}{10}-\frac{2}{10}\log_2\frac{2}{10}$*=1.522;*

Gain Ratio(T, Income)= $\frac{Gain(T,Income)}{Split\,Info(T,Income)}$ *=0.5;*

Gain Ratio(T, Property)= $\frac{Gain(T,Property)}{Split\,Info(T,Property)}$*=0.514;*

Gain Ratio(T, Unpaid-credit)=$\frac{Gain(T,Unpaid-credit)}{Split\,Info(T,Unpaid-credit)}$*=0.439;*

The 'Property' has the highest gain ratio. So, it will be selected as the root of the decision tree. Continuing the same process, the final decision tree

will be presented by (see Figure 2.2):

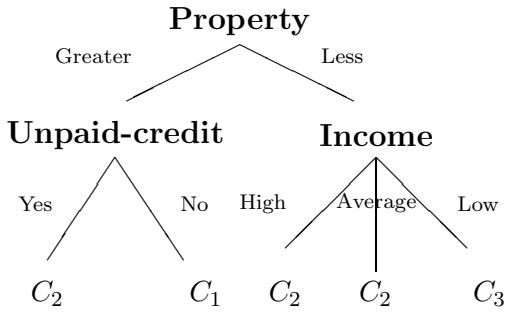

Figure 2.2: Final decision tree

Assume we have to classify the instance i: (high, less, no) using the induced decision tree. We reach a leaf labeled by C_2 which corresponds to the class of the instance i and which means that this client i will have only a part of the loan.

2.3 Pruning decision trees

2.3.1 Definition

When a decision tree is built, many of branches will reflect anomalies in the training data due to noise and outliers. Tree pruning methods address this problem of overfitting the data. Such methods typically use statistical measure to remove the least reliable branches, when the tree overfits the training cases by modeling its noise, pruning can often improve its accuracy on unseen cases. Pruning is intended also to improve comprehensibility of decision trees. These dual goals are often correlated.

"How does tree pruning work?" there are two common approaches to tree pruning.

2.3.2 Pre-pruning approach

Stop growing tree at some point during construction when it is determined that there is not enough data to make reliable choices. Pre-pruning methods

simplify decision trees by halting the induction of a complete tree using trivial stopping criterion. Upon halting, the node becomes a leaf. The leaf may hold the most frequent class among the subset of training samples at the node. The simplest pre-pruning algorithm imposes a minimum threshold on test selection measure.

2.3.3 Post-pruning approach

Grow the full tree and then remove nodes that seem not to have sufficient evidence. Post-pruning algorithms input an unpruned tree T and output pruned tree T', formed by removing one or more sub-trees from T. Pruning replaces a sub-tree with a leaf, transforming an internal node into a leaf node. Several post-pruning methods have been introduced in the literature, including minimal cost-complexity pruning (Breiman et al., 1984), minimum error pruning (Niblett & Bratko, 1986), pessimistic error pruning (Quinlan, 1987), reduced error pruning (Quinlan, 1987), critical error pruning (Mingers, 1987) and error-based pruning (Quinlan, 1993).

Minimal Cost-Complexity Pruning (MCCP): The MCCP, was developed by Breiman et al (1984). This method is also known as the CART pruning algorithm. It consists of two steps :

1. Generating a series of increasingly pruned trees $\{T_0, T_1, T_2, \ldots, T_n\}$.
2. Selecting the best tree with the lowest error rate on separate test set.

As regarding the first step, T_{i+1} is obtained from T_i by pruning all the nodes having the lowest increase in error rate per pruned leaf denoted α.

$$\alpha = \frac{R(t) - R(Tt)}{NT - 1} \tag{2.6}$$

- R(t) is the error rate if the node is pruned, which becomes a leaf belonging to only one class. It is the proportion of training examples which not belong to this class.
- R(Tt) is the error rate if the node is not pruned. It represents the average of the error rates at the leaves weighted by the number of examples at each leaf.

Pruning leads to an increase in the error rate in the training set. Dividing this increase by the number of leaves gives the increase in error rate per pruned leaf denoted α.

The method works as follows:

1. Compute α for each (non-terminal) node (except the root) in T_i.
2. Prune all the nodes with the smallest value of α, so obtaining the tree T_{i+1}.
3. Repeat this process until only root is left yields a series of pruned tree.
4. The next step is to select one of these as the final tree. The criterion for selection of the final tree is the lowest mis-classification rate on independent data set. This selection is based only on testing set accuracy.

Critical Value Pruning (CVP): This post-pruning method, proposed by Mingers (1987), is very similar to a pre-pruning technique. It consists of two steps.

1. First, a threshold is specified, named critical value. An internal node of the tree is pruned if the value obtained by the selection measure does not exceed the threshold. This node is considered as no important, because the value of the selection measure obtained in the tree creation stage estimates its strength. Repeating this procedure using increasing critical values yields a series of pruned trees
2. Second step, choose the best tree in the same way as for MCCP.

Pessimistic Error Pruning (PEP): This pruning method, proposed by Quinlan (1987), is characterized by using the same training set for both growing and pruning a tree. The mis-classification rate produced by a tree on its training set is overly optimistic: for pruning, produce overly large tree. For this reason, Quinlan introduces the continuity correction for the binomial distribution that provides "a more realistic error rate".

Minimum Error Pruning (MEP): This post-pruning method was proposed by Niblett and Bratko (1987). It consists to find the single tree, which

should theoretically, give the minimum expected error rate when classifying independent data set. This does not mean that the testing set is used, but the authors intend to estimate the error rate for unseen cases. The result should be a pruned tree that minimizes the expected error rate in classifying independent data. At each node, calculate the expected error rate (if the node is pruned and if not pruned). If pruning of the node leads to a greater expected error rate, then keep the sub-tree.

The procedure produces only a single tree; this is a drawback for the expert systems. It is improved by Cestink and Brakto MEP-2 (1991).

Reduced Error Pruning (REP): This method was proposed by Quinlan (1987), is the simplest. It can reduce the series of pruned trees by using the testing set directly, rather than using it on selection of the best tree. The method works as follows:

1. Start with complete tree and run the testing set.
2. For each internal node t, compare the number of classification errors n(Tt) made on testing set if the node is kept with the number of classification errors n(t) made when t is turned into a leaf and associated with the best class.
3. From the all the nodes, choose the one with the largest difference to prune.
4. Continue this process, until the pruning would increase the miss classification rate.

This approach generates a set of trees, ending with the smallest minimum-error tree on the testing set.

Error Based Pruning (EBP): This method was proposed by Quinlan (1993). It is considered an improvement on the PEP method, since it is based on a far more pessimistic estimate of the expected error rate. Both methods use information in the training set for building and simplifying trees. The true novelty is that EBP simplifies a decision tree T by grafting a branch Tt into the place of the parent of t itself, in addition to pruning nodes. Grafting: An internal node of decision tree is removed and replaced by one of its sub-trees.

2.4 Decision trees under uncertainty: Belief decision trees

In addition to pruning, another enhacement of standard decision trees like the other classification techniques, do not well perform their classification task in an environment characterized by uncertainty. In order to overcome this limitation, many researches have been done to adapt standard decision tree to this kind of environment. The idea was to introduce theories could represent uncertainty. Several kinds of decision trees were developed: **probabilistic decision trees** (Quinlan, 1987, 1990), **fuzzy decision trees** (Umano et al., 1994), (Zeidler & Schlosser, 1996), (Marsala, 1998), **belief decision trees** (Elouedi et al., 2000)(Elouedi et al., 2001), (Denoeux & Skarstien-Bjanger, 2000) and **possibilistic decision trees** (Ben Amor et al., 2004), (Jenhani et al., 2005).

In our work we will focus on belief decision trees. So, in this section we will present the basic concepts of BDT.

2.4.1 Definition

A belief decision tree is a decision tree in uncertain environment where the uncertainty is represented by the Transferable Belief Model (TBM) one representation of the belief function theory.

Contrary to a classical decision tree where classes and attribute values of objects are known with certainty, in a belief decision tree these two fundamental parameters may be uncertain. Such uncertainty can appear either in the construction or the classification phase.

2.4.2 Structure of training set

The structure of the training set allowing to induce a belief decision tree is different from the traditional one. The values of the attributes of each training object are known with certainty, whereas its corresponding class is uncertain. The uncertainty on classes of a training object is represented by a basic belief assignment defined on the set of possible classes. Belief decision trees use the following notations:

- T: a given training set composed by p objects I_j (j=1,...p),
- S: a set of objects belonging to the training set T,
- A: an attribute,
- Θ=$\{C_1, C_2, .., C_n\}$: the frame of discernment made of the n possible classes related the classification problem.
- $m^{\Theta}\{I_j\}$(C): the bbm given to the hypothesis that the actual class of object I_j belongs to the C $\subseteq \Theta$.

Example 2.3 *We assume there are eight objects I_j that may belong to one of the three possible classes $\{C_1, C_2, C_3\}$. The training set instances are characterized by three symbolic attributes (Income, Property and Unpaid-credit).*

The structure of the training set can be defined as follows (see Table 2.2):

Table 2.2: Training set T relative to a belief decision tree

Income	Property	Unpaid-credit	Class
High	Greater	Yes	$m^{\Theta}\{I_1\}$
Average	Less	No	$m^{\Theta}\{I_2\}$
High	Greater	Yes	$m^{\Theta}\{I_3\}$
Average	Greater	Yes	$m^{\Theta}\{I_4\}$
Low	Less	Yes	$m^{\Theta}\{I_5\}$
No	Less	No	$m^{\Theta}\{I_6\}$
High	Greater	No	$m^{\Theta}\{I_7\}$
Average	Less	Yes	$m^{\Theta}\{I_8\}$

For each object I_j belonging to the training set T, we assign a bba $m^{\Theta}\{I_j\}$ expressing beliefs on its assigned classes. These functions are defined on the frame of discernment $\Theta = \{C_1, C_2, C_3\}$.

$m^{\Theta}\{I_1\}(C_1)$=0.7; $m^{\Theta}\{I_1\}(\Theta)$=0.3;
$m^{\Theta}\{I_2\}(C_2)$=0.5; $m^{\Theta}\{I_2\}(C_1 \cup C_2)$=0.4; $m^{\Theta}\{I_2\}(\Theta)$=0.1;
$m^{\Theta}\{I_3\}(C_1)$=0.6; $m^{\Theta}\{I_3\}(\Theta)$=0.4;
$m^{\Theta}\{I_4\}(C_2)$=0.6; $m^{\Theta}\{I_4\}(C_3)$=0.3; $m^{\Theta}\{I_4\}(\Theta)$=0.1;
$m^{\Theta}\{I_5\}(C_3)$=0.7; $m^{\Theta}\{I_5\}(C_2 \cup C_3)$=0.2; $m^{\Theta}\{I_5\}(\Theta)$=0.1;

$m^{\Theta}\{I_6\}(C_3)$=0.95; $m^{\Theta}\{I_6\}(\Theta)$=0.05;
$m^{\Theta}\{I_7\}(C_1)$=0.95; $m^{\Theta}\{I_7\}(\Theta)$=0.05;
$m^{\Theta}\{I_8\}(C_2)$=0.4; $m^{\Theta}\{I_8\}(C_3)$=0.4; $m^{\Theta}\{I_1\}(\Theta)$=0.2;

2.4.3 Belief decision tree parameters

In this section, we define the major parameters leading to the construction of the belief decision tree.

Attribute selection measures in a belief decision tree

The major parameter ensuring the building of a decision tree is the attribute selection measure allowing to determine the attribute to assign to the decision node of the induced belief decision tree at each step. There are two attribute selection measures (Elouedi et al., 2000)(Elouedi et al., 2001):

1. The first one is an extension of the classical approach developed by Quinlan and based on the gain ratio criterion. It is called the averaging approach.

2. The second one represents ideas behind the TBM it self and based on a distance criterion. It is called the conjunctive approach.

The averaging approach

Under this approach, the attribute selection measure is based on the entropy computed from the average pignistic probabilities computed from the pignistic probabilities of each instance in the node. The steps to choose the appropriate attribute are represented as follow:

1. Compute the pignistic probability of each object I_j by applying the pignistic transformation to $m^{\Theta}\{I_j\}$.

2. Compute the average pignistic probability function $BetP^{\Theta}\{\mathrm{S}\}$ taken over the set of objects S. For each $C_i \in \Theta$,

$$BetP^{\Theta}\{S\}(C_i) = \frac{1}{|S|} \sum_{I_j \in S} BetP^{\Theta}\{I_j\}(C_i) \tag{2.7}$$

3. Compute the entropy Info(S) of the average pignistic probabilities in the set S. This Info(S) value is equal to:

$$Info(S) = -\sum_{i=1}^{n} BetP^{\Theta}\{S\}(C_i) \log_2 BetP^{\Theta}\{S\}(C_i) \tag{2.8}$$

4. Select an attribute A. Collect the subset S_v^A made with cases of S having v as a value for the attribute A. Then, compute the average pignistic probability for objects in subset S_v^A. let the result be denoted $BetP^{\Theta}\{S_v^A\}$.

5. Compute $Info_A(S)$, as Quinlan:

$$Info_A(S) = \sum_{v \in D(A)} \frac{|S_v^A|}{|S|} Info(S_v^A) \tag{2.9}$$

   where D(A) is the domain of the possible values of the attribute A and Info(S_v^A) is computed from the equation(2.8) using $BetP^{\Theta}\{S_v^A\}$.

6. Compute the information gain provided by the attribute A in the set of objects S such that:

$$Gain(S,\, A) = Info(S) - Info_A(S) \tag{2.10}$$

7. Using the split Info, compute the gain ratio relative to attribute A:

$$Gain\, Ratio(S,\, A) = \frac{Gain(S, A)}{SplitInfo(S, A)} \tag{2.11}$$

   Where

$$Split\, Info(S, A) = -\sum_{v \in D(A)} \frac{|S_v^A|}{|S|} \log_2 \frac{|S_v^A|}{|S|} \tag{2.12}$$

8. Repeat for every attribute A belonging to the set of attributes that can be selected. Next, choose the one that maximizes the gain ratio.

Example 2.4 *Let's illustrate this attribute selection measure by an example already presented (see Example 2.3).*

We have, to compute the average pignistic probability relative to the training set T (see Table 2.3):

Table 2.3: Computation of $BetP^{\Theta}\{T\}$

	C_1	C_2	C_3
$BetP^{\Theta}\{I_1\}$	0.8	0.1	0.1
$BetP^{\Theta}\{I_2\}$	0.23	0.73	0.04
$BetP^{\Theta}\{I_3\}$	0.74	0.13	0.13
$BetP^{\Theta}\{I_4\}$	0.04	0.63	0.33
$BetP^{\Theta}\{I_5\}$	0.04	0.13	0.83
$BetP^{\Theta}\{I_6\}$	0.02	0.02	0.96
$BetP^{\Theta}\{I_7\}$	0.96	0.02	0.02
$BetP^{\Theta}\{I_8\}$	0.06	0.47	0.47
Mean = $BetP^{\Theta}\{T\}$	0.36	0.28	0.36

These probabilities will be used to compute Info(T) relative to the whole training set T (see equation 2.8):

$$Info(T) = -\sum_{i=1}^{3} BetP(C_i) \log_2 BetP(C_i) = 1.575;$$

Once the Info(T) is calculated, we have to look for the $Info_{Income}(T)$, $Info_{Property}(T)$ *and* $Info_{Unpaid-credit}(T)$.

Let us do the computation for the income attribute (see table 2.4):

Table 2.4: Computation of $BetP^{\Theta}\{T^{Income}_{High}\}$, $BetP^{\Theta}\{T^{Income}_{Average}\}$, $BetP^{\Theta}\{T^{Income}_{Low}\}$ and $BetP^{\Theta}\{T^{Income}_{No}\}$

	C_1	C_2	C_3
$BetP^{\Theta}\{T^{Income}_{High}\}$	0.84	0.08	0.08
$BetP^{\Theta}\{T^{Income}_{Average}\}$	0.11	0.61	0.28
$BetP^{\Theta}\{T^{Income}_{Low}\}$	0.04	0.13	0.83
$BetP^{\Theta}\{T^{Income}_{No}\}$	0.02	0.02	0.96

The next step consists in the computation of $Info_{Income}(T)$. By applying the equation (2.9), we get:

$$Info_{Income}(T) = 0.921;$$

Then, we compute the information gain (see equation 2.10), we get respectively:

$$Gain(T, Income) = Info(T) - Info_{Income}(T) = 0.654;$$
$$Split\,Info(T,\ Income) = 1.811;$$
$$Gain\,Ratio(T,\ Income) = 1.361;$$

By applying the same process, we get:

Gain Ratio(T, Proverty)=0.276;
Gain Ratio(T, Unpaid-credit)=0.004;

The attribute that maximizes the gain ratio is the income attribute. It will be chosen as the root of the belief decision tree.

The conjunctive approach

The conjunctive approach is based on an intra-group distance quantifying for each attribute value how strongly objects are close from each others. The different steps upon this attribute selection measure ensuring the building of a decision tree are the following ones:

1. For each training object, compute:

$$K^{\Theta}\{Ij\}(C) = -\ln q^{\Theta}\{I_j\}(C)\,\forall C \subseteq \Theta \tag{2.13}$$

   from the bba $m^{\Theta}\{I_j\}$.

2. For each attribute value v of an attribute A, compute the joint $K^{\Theta}\{S_v^A\}$ defined on Θ, the set of possible classes by:

$$K^{\Theta}\{S_v^A\} = \sum_{I_j \in S_v^A} K^{\Theta}\{I_j\} \tag{2.14}$$

3. For each attribute value, the intra-group distance $\mathrm{SumD}(S_v^A)$ is defined by:

$$SumD(S_v^A) = \frac{1}{|S_v^A|} \sum_{I_j \in S_v^A} \sum_{X \subseteq \Theta} (K^{\Theta}\{I_i\}(X) - \frac{1}{|S_v^A|} K^{\Theta}\{S_v^A\}(X))^2 \tag{2.15}$$

4. Compute $SumD_A(S)$ representing the weighted sum of the different SumD(S_v^A) relative to each value v of the attribute A:

$$SumD_A(S) = \sum_{v \in D(A)} \frac{|S_v^A|}{|S|} SumD(S_v^A) \tag{2.16}$$

5. By analogy to our averaging approach, we may also compute Diff(S,A) defined as the difference between SumD(S) and $SumD_A(S)$:

$$Diff(S, A) = sumD(S) - SumD_A(S) \tag{2.17}$$

Where

$$SumD(S) = \frac{1}{|S|} \sum_{I_j \in S} \sum_{X \subseteq \Theta} (K^{\Theta}\{I_i\}(X) - \frac{1}{|S|} K^{\Theta}\{S\}(X))^2 \tag{2.18}$$

6. Using the Split Info, compute the diff ratio relative to the attribute A:

$$Diff\ Ratio(S, A) = \frac{Diff(S, A)}{SplitInfo(S, A)} \tag{2.19}$$

7. For every attribute repeat the same process, and choose the one that maximizes the diff ratio.

Example 2.5 *Let's use the same training set presented in the example and let's try to apply our attribute selection measure based on this conjunctive approach in order to find the test attribute.*

We start by computing SumD(T), the sum of distances separating each training instance to the whole set T. We have:

$$m\{T\} = m^{\Theta}\{I_1\} \cap m^{\Theta}\{I_2\} \cap ... \cap m^{\Theta}\{I_8\}$$

$$SumD(T) = \frac{1}{8} \sum_{I_j \in T} \sum_{C \subseteq \Theta} (K^{\Theta}\{I_j\}(C) - \frac{1}{8} K^{\Theta}\{S\}(C))^2$$

$$= 13.437;$$

We proceed with the computation for the income attribute. Four values maybe possible which are high, average, low and no income.

$$SumD_{Income}(T) = \frac{|T_{High}^{Income}|}{|T|} SumD(T_{High}^{Income}) + \frac{|T_{Average}^{Income}|}{|T|} SumD(T_{Average}^{Income})$$

$$+\frac{|T_{Low}^{Income}|}{|T|} SumD(T_{Low}^{Income}) + \frac{|T_{No}^{Income}|}{|T|} SumD(T_{No}^{Income}) = 2.782;$$

By applying the same process for the other attributes, we get:

$$SumD_{Property}(T) = 7.730;$$

$$SumD_{Unpaid-credit}(T) = 9.353;$$

We have also to compute Diff(T,A) for A $\in$ {*Income, Property, Unpaid-credit*}*. We get:*

Diff(T, Income)=10.655;
Diff(T, Property)=5.707;
Diff(T, Unpaind-credit)=4.084;

Then, the computation of the Diff ratio gives the following results:

Diff Ratio(T, Income)=5.882;
Diff Ratio(T, Property)=5.707;
Diff Ratio(T, Unpaind-credit)=5.279;

The application of the Diff Ratio criterion leads to the choice of the income attribute as the test attribute relative to the training set T.

Partitioning strategy

The partitioning strategy for the construction of a belief decision tree is similar to the partitioning strategy used in the classical tree.

Stopping criteria

Four strategies are proposed as stopping criteria:

1. If the treated node includes only one instance.
2. If the treated node includes only instances for which the $m^{\Theta}Ij$'s are equal.

3. If all the attributes are split.

4. If the value of the applied attribute selection measure (using either the gain ratio or the diff ratio) for the remaining attributes is less or equal than zero.

Structure of leaves

Each leaf in the induced tree will be characterized by a bba. According to the used attribute selection measure:

- Using the averaging approach, the leaf's bba is equal to the average of the bba's of the objects belonging to this leaf.
- Using the conjunctive approach, the leaf's bba is the result of combination of the bba's of objects belonging to this leaf.

2.4.4 Belief decision tree procedures

Like the standard decision tree, the belief decision tree is composed of two principal procedures: the building or the construction of the tree from uncertain data and the classification of new instances that may be characterized by uncertain or even missing attribute values.

Building procedure

Building a decision tree in this context of uncertainty will follow the same steps presented in C4.5 algorithm. Furthermore, this algorithm is generic since it offers two possibilities for selecting attributes by using either the averaging approach or the conjunctive one.

Example 2.6 *Let's continue with the examples proposed in example 2.4 Our objective is to generate the two belief decision trees respectively to the average approach and the conjunctive approach.*

- Averaging approach: As computed in the example 2.4, we have found that:
  Gain Ratio(T, Income)=0.361;
  Gain Ratio(T, Property)=0.276;

Gain Ratio(T, Unpaid-credit)=0.004;

We choose the income attribute as the root of the belief decision tree relative to the training set T, since it presents the highest gain ratio.

The final belief decision tree induced by our algorithm using the averaging approach is given by (see Figure 2.3):

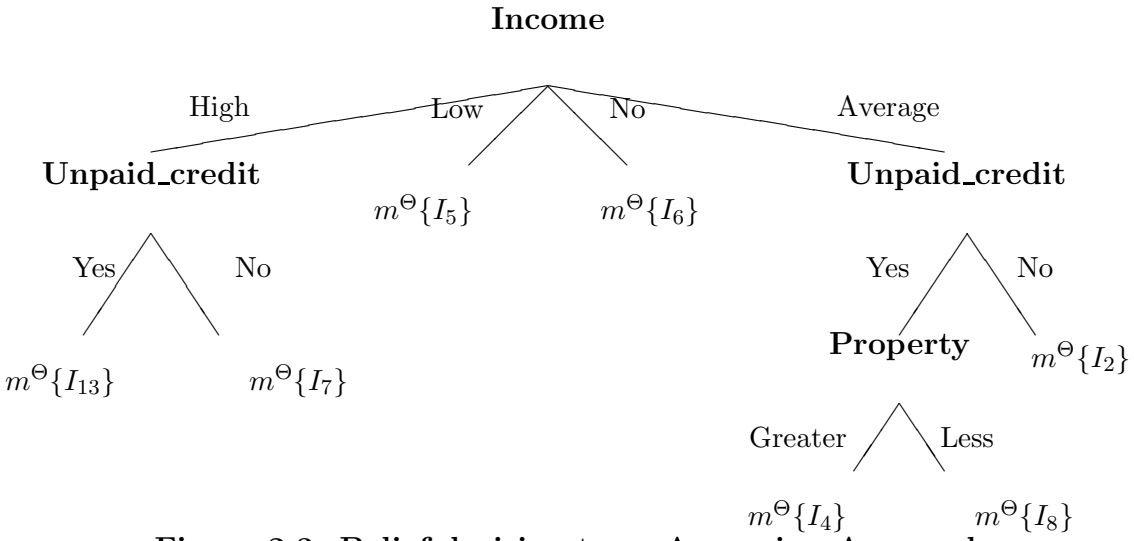

Figure 2.3: Belief decision tree: Averaging Approach

Where $m^{\Theta}\{I_{13}\}$ is the average bba relative to the objects $\{I_1\}$ and $\{I_3\}$. So, we get:
$m^{\Theta}\{I_{13}\}(C_1)$=0.65;

$m^{\Theta}\{I_{13}\}(\Theta)$=0.35;

- *Conjunctive approach: As computed in example 2.4, we have found that:*

Diff Ratio(T, Income)=5.882;
Diff Ratio(T, Property)=5.707;
Diff Ratio(T, Unpaid-credit)=4.279;

As noted, the income attribute presents the highest value of the Diff Ratio. Hence, it would be chosen as the root of the belief decision tree (like with the averaging approach).

By applying the same process to the training subsets induced, the final belief decision tree is (see Figure 2.4):

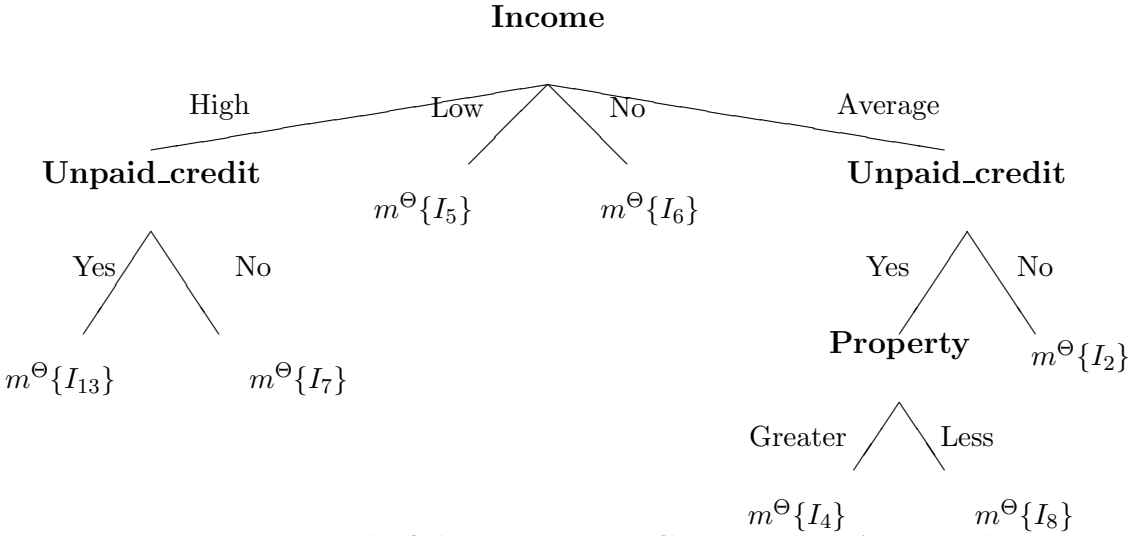

Figure 2.4: Belief decision tree: Conjunctive Approach

Where $m^{\Theta}\{I_{13}\}$ *is defined as follows:*
$m^{\Theta}\{I_{13}\} = m^{\Theta}\{I_1\} \cap m^{\Theta}\{I_3\}$
So, we get:
$m^{\Theta}\{I_{13}\}(C_1)$*=0.88;*
$m^{\Theta}\{I_{13}\}(\Theta)$*=0.12;*

Classification procedure

Once the belief decision tree is constructed, it is able to classify an object described by an exact value for each one of its attribute, we have to start from the root of the belief decision tree, and repeat to test the attribute at each node by taking into account the attribut value until reaching a leaf.

As a leaf is characterized by a bba on classes, the pignistic transformation is applied to get the pignistic probability on the classes of the object to classify in order to decide its class. For instance, one can choose the class

having the highest pignistic probability.

Belief decision trees deal also with the classification of new instances characterized by uncertainty in the values of their attributes. The idea to classify such objects is to look for leaves that the given instance may belong to by tracing out possible paths induced by the different attribute values of the object to classify.

2.5 Conclusion

Decision trees are considered as one of the best technique for classification tasks in machine learning. So, it is applied in many new computer science applications, notably those related to knowledge discovery.

In our work, we will focus on belief decision trees. This new approach is based on both the decision tree technique and the belief function theory in order to cope with uncertainty.

In this chapter, we have presented the two major procedures related to the belief decision tree technique. We have described the different steps of the procedure allowing the construction of the tree from uncertain data.

Inducing a decision tree, and consequently a belief decision tree, without applying any mechanism of pruning leads in most cases to very large trees with many nodes and leaves. So, our objective is to develop post-pruning methods in order to simplify the belief decision tree and this will be detailed in the next chapter.

Part II

Pruning Belief Decision Trees

Chapter 3

Pruning belief decision tree methods

3.1 Introduction

As mentioned in chapter 2, a belief decision tree is a decision tree developed in order to handle uncertainty about the actual class of the objects in the training set and also to classify objects characterized by uncertain attributes. It is a classification technique based on decision trees within the framework of belief function theory.

Inducing a belief decision tree with averaging or conjunctive approaches, may lead in most cases to very large trees with bad classification accuracy. Several pruning methods have been developed to cope with is problem. All these methods deal with only standard decision trees. So, our objective is to develop pruning methods in order to simplify the belief decision tree and improve its classification accuracy.

This chapter is dedicated to the presentation of our pruning belief decision tree methods in averaging and conjunctive approaches.

3.2 Comparison between post-pruning methods

In this work, we focused on post-pruning approach. Pre-pruning in belief decision tree is developed by Elouedi et al. (2002b) by improving the stopping criteria concerning the value of the selection measure. However, all standard post-pruning methods deal with standard decision trees and not with the other variants. So, we suggest is to adapt one of these methods to belief decision tree in both approaches averaging and conjunctive.

In this section, we will give some critical comments on the strengths and weaknesses of the different post-pruning methods, based on the work of Mingers (1989) and Esposito et al. (1997). They review and compare many of existing post-pruning methods over several datasets. Their studies concluded that there were some differences between them in terms of complexity and classification accuracy.

Table 3.1 gives a comparative summary of standard post-pruning methods (E refers to Esposito et al. and M refers to Mingers). From this table we will choose the method to adapt to simplify the belief decision tree.

Table 3.1: A comparative summary of post-pruning methods

Methods	Benefits	Limitations
MCCP (M,E)	Accurate(M,E) Good tree size (E)	
REP (M,E)	Accurate(M, but not E) Good tree size (E)	
EBP (E)	Very Accurate	Underprunes
MEP (M)		Underprunes High complexity; Sensitive to number of classes; Low accuracy
CVP (M,E)		Low accuracy (M,E) Underprunes(E)
PEP (M,E)	Accurate (E but not M)	Underprunes(M but not E)

This table shows that minimum error pruning (MEP), critical value pruning (CVP) and pessimistic error pruning (PEP) are weak pruning methods in terms of size pruned tree and accuracy. Besides, they have several problems.

In MEP, the number of classes strongly affects the degree of pruning, leading to unstable results. It assumes uniform class distribution. It is seldom true in practice. It produces only a single tree. It is a drawback for the expert.

In CVP, The degree of pruning clearly changes with the critical value: the choice of a high critical value results great degree of pruning and small resulting over-pruned tree, leading to unstable results.

PEP does not produce a selection of trees. The corrected mis-classification estimate is optimistic. So, it leads overly large tree. PEP uses a top-down pruning strategy, which encounters the same problem as pre-pruning.

However, minimal cost-complexity pruning (MCCP), reduced error pruning (REP) and error based pruning (EBP) perform well.

In our work, we will choose MCCP to adapt for pruning belief decision tree. This pruning method is appealing because it takes into account both the classification error and the tree complexity with its measure α. It also produces a selection of trees for the expert to study, where it is helpful if several trees, pruned to different degree are available.

However, EBP and REP produce a single tree. In addition, EBP is underprune method (no important reduction in size). It is a disadvantage for the pruning of the belief decisison tree.

3.3 MCCP in standard case

As explained in chapter 2, MCCP consists of two phases.

1. The pruning phase consists in generating a series of increasingly pruned trees $\{T_1, T_2, ..., T_n\}$. T_{i+1} is obtained from T_i by pruning all the nodes have the lowest α.

2. The section phase consists in choosing the best tree with lowest error rate on testing set.

Example 3.1 *To explain how to compute α in certain case, consider the node N in Figure 3.1 of a decision tree induced from a training set of 44 objects and 3 classes. The node N contains 8 objects (three objects of class C_1, four objects of class C_2 and one object of class C_3) and has 2 leaves. The first belongs to class C_1 and the second belongs to class C_2.*

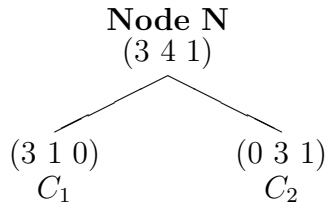

Figure 3.1: Unpruned Node

The error rate if the node is not pruned is the sum of error rates at leaves weighted by the number of examples at each leaf:

$$R(Tt) = \frac{1}{4} * \frac{4}{44} + \frac{1}{4} * \frac{4}{44} = \frac{1}{22}$$

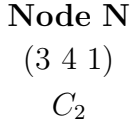

Figure 3.2: Pruned Node

In the Figure 3.2, the node N becomes a leaf of class C_2 by pruning it and the leaf is allocated to the class which occurs most frequently. So, the error rate of leaf is then the proportion of training examples which do not belong to that class:

$$R(t) = \frac{4}{8} * \frac{8}{44} = \frac{1}{11}$$

The increase in error rate per pruned leaf, denoted α:

$$\alpha = \frac{R(t) - R(Tt)}{NT - 1} = \frac{\frac{1}{22}}{2 - 1} = \frac{1}{22}$$

If the node N has the lowest α, starting pruning it.

3.4 Pruning belief decision trees

In this section we will propose our pruning belief decision tree methods in averaging and conjunctive approaches based on standard minimal cost-complexity pruning (MCCP).

3.4.1 MCCP in Averaging Approach

Our objective is to develop a pruning method based on standard minimal cost-complexity pruning to prune belief decision tree in averaging approach. To prune a node in MCCP, we compute the error rate if the node is pruned or not. To do this, we must know at each node or leaf, the number of objects belonging to each class. However, in a belief decision tree, the class of the objects are represented by a basic belief assignment and not by a certain class.

The idea is to use the pignistic transformation. It is a function which can transform the belief function to probability function in order to make decisions from beliefs. This function is used to build the belief decision tree in averaging approach. So, we will transform beliefs on classes to a distribution of probability at each node or leaf to know the proportion of each class.

In this section, we propose the following steps to prune the belief decision tree by adapting MCCP.

1. For each node in the belief decision tree, compute the pignistic probability of each object I_j by applying the pignistic transformation to $m^{\Theta}\{I_j\}$.

2. Compute the sum pignistic probability function $BetP^{\Theta}\{\mathrm{S}\}$ taken over the set of objects S belong to a node N. For each $C_i \in \Theta$,

$$BetP^{\Theta}\{S\}(C_i) = \sum_{I_j \in S} BetP^{\Theta}\{I_j\}(C_i)$$

$BetP^{\Theta}\{\mathrm{S}\}$ represents the number of objects belong to each class $C_i \in \Theta$ for a node N.

In this way, we can compute the number of error of each node. It is the sum of objects not allocated to the class which occurs most frequently.

3. Compute the error rate if the node is pruned, which becomes a leaf.

$$R(t) = \frac{\sum_{C_i \in \Theta} \left(BetP^{\Theta}\{S\}(C_i)\right) - Max\left(BetP^{\Theta}\{S\}(C_i)\right)}{|T|} \tag{3.1}$$

$Max\left(BetP^{\Theta}\{S\}(C_i)\right)$ represents the number of training objects belong to the class which occurs most frequently.

4. Compute the error rate if the node is not pruned.

$$R(T_t) = \sum R(i) \text{ for i = sub-tree leaves} \tag{3.2}$$

5. Compute the increase in error per pruned leaf, denoted α_ave.

$$\alpha\_ave = \frac{R(t) - R(Tt)}{NT - 1} \tag{3.3}$$

6. Repeat the same process for every node in the belief decision tree only the root.

If a node has the lowest α_ave, starting pruned it and obtain the first pruned tree. The node becomes a leaf represented by the average bba relative to the objects belong to it.

Continue this process until the root is left, yields a series of pruned trees.

In the selection phase, from the series of pruned trees select the best tree with the lowest error rate on testing set.

Example 3.2 *To explain how our pruning method works, we take the Table 3.2 showing the training set used to build the belief decision tree and the Figure 3.3 showing the final belief decision tree in averaging approach inducing from this training set.*

Table 3.2: Training set T relative to a belief decision tree

Income	Property	Unpaid-credit	Class
High	Greater	Yes	$m^{\Theta}\{I_1\}$
Average	Less	No	$m^{\Theta}\{I_2\}$
High	Greater	Yes	$m^{\Theta}\{I_3\}$
Average	Greater	Yes	$m^{\Theta}\{I_4\}$
Low	Less	Yes	$m^{\Theta}\{I_5\}$
No	Less	No	$m^{\Theta}\{I_6\}$
High	Greater	No	$m^{\Theta}\{I_7\}$
Average	Less	Yes	$m^{\Theta}\{I_8\}$

$m^{\Theta}\{I_1\}(C_1)$=0.7; $m^{\Theta}\{I_1\}(\Theta)$=0.3;
$m^{\Theta}\{I_2\}(C_2)$=0.5; $m^{\Theta}\{I_2\}(C_1 \cup C_2)$=0.4; $m^{\Theta}\{I_2\}(\Theta)$=0.1;
$m^{\Theta}\{I_3\}(C_1)$=0.6; $m^{\Theta}\{I_3\}(\Theta)$=0.4;
$m^{\Theta}\{I_4\}(C_2)$=0.6; $m^{\Theta}\{I_4\}(C_2 \cup C_3)$=0.4;
$m^{\Theta}\{I_5\}(C_3)$=0.7; $m^{\Theta}\{I_5\}(C_2 \cup C_3)$=0.2; $m^{\Theta}\{I_5\}(\Theta)$=0.1;
$m^{\Theta}\{I_6\}(C_3)$=0.95; $m^{\Theta}\{I_6\}(\Theta)$=0.05;
$m^{\Theta}\{I_7\}(C_1)$=0.95; $m^{\Theta}\{I_7\}(\Theta)$=0.05;
$m^{\Theta}\{I_8\}(C_2)$=0.2; $m^{\Theta}\{I_8\}(C_3)$=0.5; $m^{\Theta}\{I_8\}(\Theta)$=0.3;

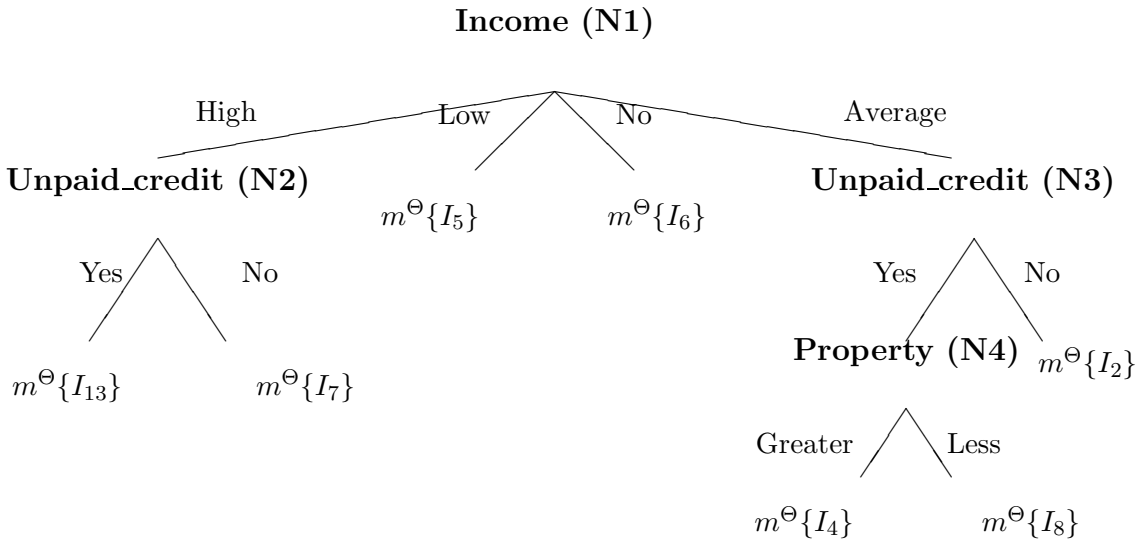

Figure 3.3: Belief decision tree: Averaging Approach

The node N1 contains 8 objects. Each object is represented by a bba m. So, we suggest in step 1 of the algorithm to compute the pignistic probability of each object I_j by applying the pignistic transformation to $m^{\Theta}\{I_j\}$.

Then in step 2 of the algorithm, do the sum of the pignistic probability taken over the set of objects belong to the node N1 for each $C_i \in \Theta$. So, we obtain the table 3.3.

Table 3.3: Computation of $BetP^{\Theta}\{T\}$

	C_1	C_2	C_3
$BetP^{\Theta}\{I_1\}$	0.8	0.1	0.1
$BetP^{\Theta}\{I_2\}$	0.23	0.73	0.04
$BetP^{\Theta}\{I_3\}$	0.74	0.13	0.13
$BetP^{\Theta}\{I_4\}$	0	0.8	0.2
$BetP^{\Theta}\{I_5\}$	0.04	0.13	0.83
$BetP^{\Theta}\{I_6\}$	0.02	0.02	0.96
$BetP^{\Theta}\{I_7\}$	0.96	0.02	0.02
$BetP^{\Theta}\{I_8\}$	0.1	0.3	0.6
Sum = $BetP^{\Theta}\{T\}$	2.89	2.23	2.88

The node N1 has 2.89 objects of class C_1, and 2.23 of class C_2 and 2.88 of C_3.

If we continue the same process for all nodes in the belief decision tree, we obtain this tree in Figure 3.4 In this way, it is easy to prune the belief decision tree in averaging approach using minimal cost-complexity pruning method.

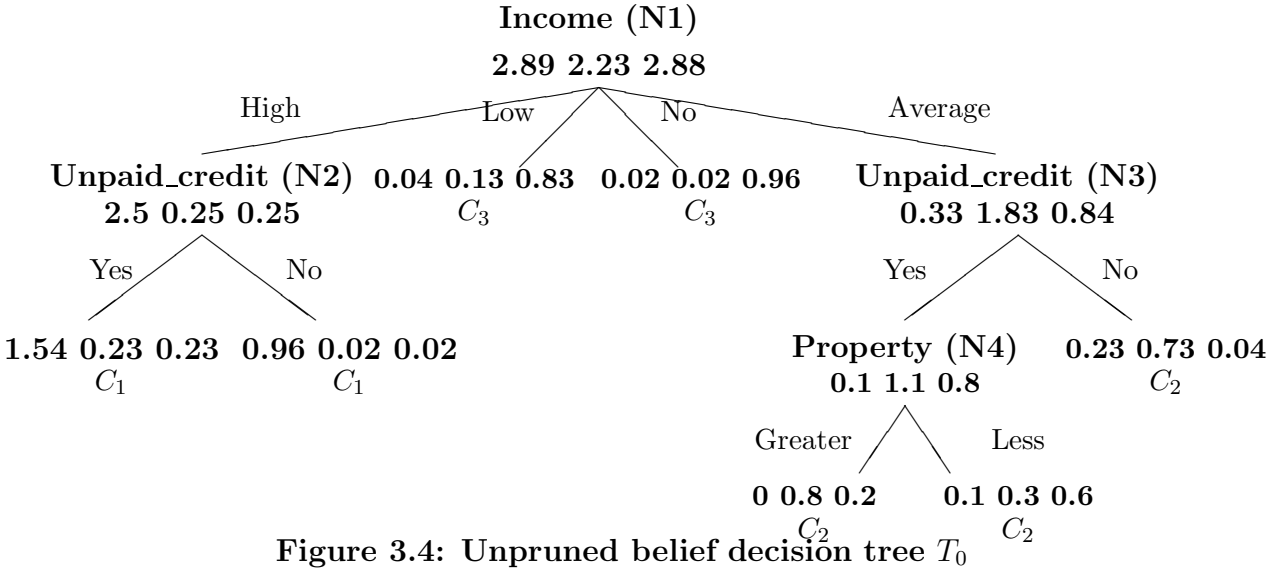

Figure 3.4: Unpruned belief decision tree T_0

Pruning phase:

Iteration 1:

For each node in T_0 only the root, compute the value of the increase per pruned leaf, denoted α_ave:

N2:

Error rate if the node is pruned (see equation 3.1):

$$R(t) = \frac{3 - 2.5}{8} = \frac{0.5}{8};$$

Error rate if the node is not pruned (see equation 3.2):

$$R(Tt) = \frac{2 - 1.54}{8} + \frac{1 - 0.96}{8} = \frac{0.5}{8};$$

Increase in error rate per pruned leaf (see equation 3.1):

$$\alpha\_ave = \frac{\frac{0.5-0.5}{8}}{2 - 1} = 0; \ (min)$$

N3:

Error rate if the node is pruned:

$$R(t) = \frac{3 - 1.83}{8} = \frac{1.17}{8};$$

Error rate if the node is not pruned:

$$R(Tt) = \frac{1 - 0.8}{8} + \frac{1 - 0.6}{8} + \frac{1 - 0.73}{8} = \frac{0.87}{8};$$

Increase in error rate per pruned leaf:

$$\alpha\_ave = \frac{\frac{1.17-0.87}{8}}{3 - 1} = 0.01875;$$

N4:

Error rate if the node is pruned:

$$R(t) = \frac{2 - 1.1}{8} = \frac{0.9}{8};$$

Error rate if the node is not pruned:

$$R(Tt) = \frac{1 - 0.8}{8} + \frac{1 - 0.6}{8} = \frac{0.6}{8};$$

Increase in error rate per pruned leaf:

$$\alpha\_ave = \frac{\frac{0.9-0.6}{8}}{2 - 1} = 0.0375;$$

The node N2 has the lowest value of α_ave. So, starting pruning it, we obtain the first pruned tree (see Figure 3.5).

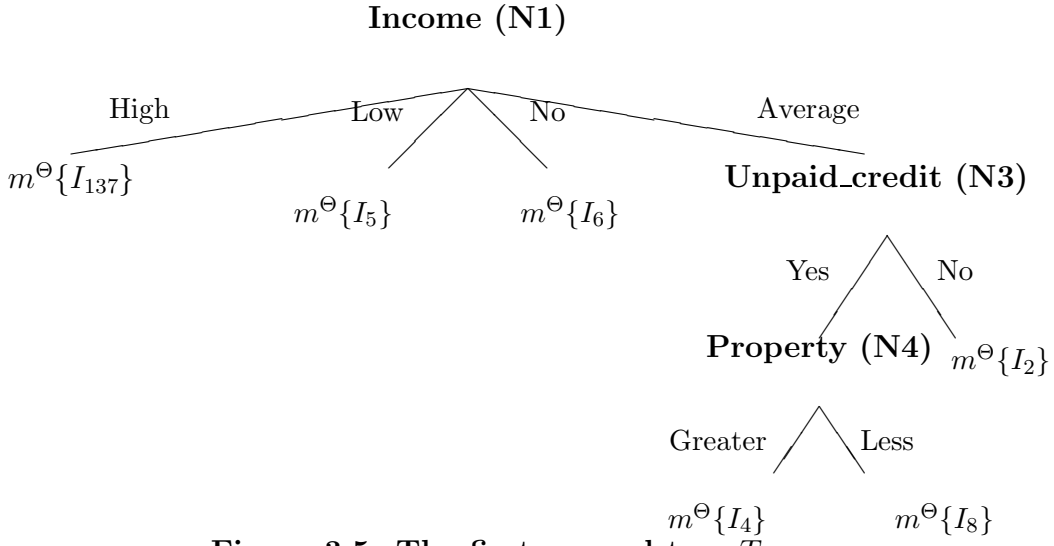

Figure 3.5: The first pruned tree T_1

So, the node N2 become a leaf represented by the average of the bba of objects in this leaf.

Iteration 2:

Continue the same process, computing the value of α_ave for all remaining nodes only the root in the tree T_1:

N3:

Error rate if the node is pruned:

$$R(t) = \frac{3 - 1.83}{8} = \frac{1.17}{8};$$

Error rate if the node is not pruned:

$$R(Tt) = \frac{1 - 0.8}{8} + \frac{1 - 0.6}{8} + \frac{1 - 0.73}{8} = \frac{0.87}{8};$$

Increase in error rate per pruned leaf:

$$\alpha\_ave = \frac{\frac{1.17-0.87}{8}}{3-1} = 0.01875;$$

N4:

Error rate if the node is pruned:

$$R(t) = \frac{2 - 1.1}{8} = \frac{0.9}{8};$$

Error rate if the node is not pruned:

$$R(Tt) = \frac{1 - 0.8}{8} + \frac{1 - 0.6}{8} = \frac{0.6}{8};$$

Increase in error rate per pruned leaf:

$$\alpha\_ave = \frac{\frac{0.9-0.6}{8}}{2 - 1} = 0.0375;$$

The node N3 has the lowest α_ave, so pruning it. We obtain the tree T_2 (see Figure 3.6).

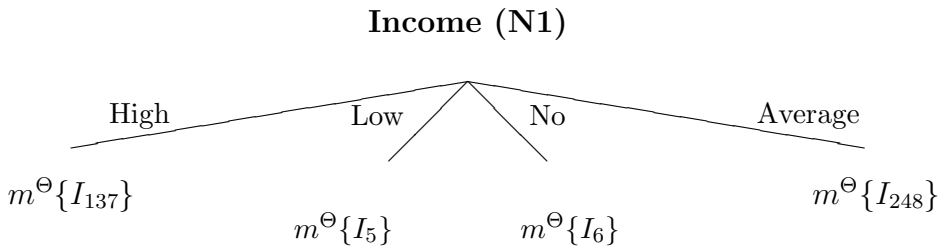

Figure 3.6: The second pruned tree T_2

Only the root is left, stopping the process.

Selection phase :

From the series of pruned trees $\{T_0, T_1, T_2\}$ select the best tree with the lowest error rate on testing set

3.4.2 MCCP in Conjunctive Approach

In this section, we will propose our second pruning belief decision tree method in conjunctive approach. This method must be should closer to the TBM itself like in the building method.

The conjunctive approach of building BDT is based on an intra-group distance quantifying for each attribute how strongly objects are close from each others.

With MCCP in a certain case, we start pruning the nodes having the lowest increase in error rate on training set if it will be pruning (nodes have the minimum α) because pruning a node leads to an increase in error rate on testing set. So by analogy, our idea is to start pruning nodes having the lowest increase of distance between objects with pruning. Because pruning will lead to an increase in the distance between objects. The distance of a node is superior than the distance of its leaves.

We propose the following steps to prune belief decision tree in conjunctive approach based on MCCP.

1. Compute the distance if the node is pruned, denoted by D(t). It represent the sum of distances separating each training instances belonging to the node. This distance between training objects is used in the attribute selection measure for the conjunctive approach (Elouedi et al., 2001).

$$D(t) = \frac{1}{|S|} \sum_{I_j \in S} \sum_{C \subseteq \Theta} (K^{\Theta}\{I_j\}(C) - \frac{1}{|S|} K^{\Theta}\{S\}(C))^2 \tag{3.4}$$

   Where

$$K^{\Theta}\{Ij\}(C) = -\ln q^{\Theta}\{I_j\}(C) \ \ \forall C \subseteq \Theta \tag{3.5}$$

2. Compute the distance if the node is not pruned, denoted by D(Tt): the sum of distance between objects in the leaves. Pruning leads to an increase in the distance between objects.

3. Dividing this increase by the number of leaves, we obtain the increase in distance by pruned leaf, denoted by α_conj.

$$\alpha\_conj = \frac{D(t) - D(Tt)}{NT - 1} \tag{3.6}$$

   Where NT is the number of leaves.

4. Repeat the same process for every node in the belief decision tree only the root.

   If a node has the lowest α_conj starting pruned it and obtain the first pruned tree. The node becomes a leaf represented by the conjunctive bba relative to the objects belong to it.

Continue this process until the root is left, yields a series of pruned trees. In the selection phase, from the series of pruned trees select the best tree with the lowest error rate on testing set.

Example 3.3 *Let's continue with the example 3.2 to explain how compute the value of α_conj for the node Unpaid-credit (N2) composed of 3 objects I_1,I_3 and I_7 and has two leaves. I_1 and I_3 belong to leaf F1 and I_7 belongs to the other leaf F2:*

The distance between objects if the node is pruned:

$$D(t) = \frac{1}{3} \sum_{I_j \in S} \sum_{C \subseteq \Theta} (K^{\Theta}\{I_j\}(C) - \frac{1}{3} K^{\Theta}\{S\}(C))^2 = 5.078;$$

The distance between objects if the node is not pruned:

$$D(Tt) = D(F1) + D(F2) = 0.1241 + 0 = 0.1241;$$

The increase in distance per pruned leaf:

$$\alpha\_conj = \frac{5.078 - 0.1241}{2 - 1} = 4.9541;$$

3.5 Conclusion

In this chapter, we have detailed our pruning belief decision tree methods in averaging and conjunctive approaches. We have adapted a standard post-pruning method MCCP in belief decision tree.

In the next chapter, we will present the implementation for checking the performance of our pruning belief decision tree methods comparing with BDT without pruning (Elouedi et al., 2001) and with pre-pruning (Elouedi et al., 2002b).

Then, we will show different results obtained from simulations and that have been performed on real databases in both certain and uncertain cases.

Chapter 4

Implementation and simulation

4.1 Introduction

Implementation our pruning belief decision tree methods in both averaging and conjunctive approaches seems imperative since it allows us have an idea concerning the performance of our proposed methods.

Once the different programs are implemented, for checking the performance of our pruning belief decision tree methods in averaging and conjunctive approaches and judging their qualities and make comparisons with BDT without pruning (Elouedi et al., 2001) and with pre-pruning (Elouedi et al., 2002b). We have performed several tests and simulations on real databases obtained from the U.C.I. repository[1]. Different results carried out from these simulations will be presented and analyzed in order to evaluate our proposed methods.

In this chapter, we will represent the different results from real databases in an uncertain context in order to evaluate the performance of our pruning methods comparing with BDT without pruning and with pre-pruning.

[1]http://www.ics.uci.edu/ mlearn/MLRepository.html

4.2 Implementation

In order to test our two methods in averaging and conjunctive approaches, we have developed programs in Matlab V6.5. Obviously, we have implemented the pruning algorithm as a well as selection algorithm that we have detailed in Chapter 3. The outputs of our programs are basically:

1. A series of pruned decision trees induced form the two methods (averaging and conjunctive)

2. The best pruned tree has the best Percent of Correct Classification, denoted PCC on testing set.

4.3 Simulations and results

4.3.1 Experimental setup

The implementation of the pruning belief decision tree algorithms will be useful in the simulation phase. We have performed several tests and simulations on real databases obtained from the U.C.I. repository. Different results carried out from these simulations will be presented and analyzed in order to evaluate our proposed methods.

4.3.2 Constructing uncertainty in training set

In classification problems, training sets are generally composed with training objects described by known attribute values and classes. Instances with partially known classes are usually eliminated from the databases, probably because the users were in pain to know what to do with these messy objects.

The belief decision trees are essentially built to handle uncertain classes where their uncertainty is represented by a bba given on the set of possible classes. So, the question is how will we construct these bba's?

These bba's are created artificially. They take into account three basic parameters:

- The real classes of the training instances.

- Degree of uncertainty
  - No uncertainty : we take $\mathbf{P}$=0
  - Low degree of uncertainty: we take $0 \prec \mathbf{P} \preceq 0.3$
  - Middle degree of uncertainty: we take $0.3 \prec \mathbf{P} \preceq 0.6$
  - High degree of uncertainty: we take $0.6 \prec \mathbf{P} \preceq 1$
- The number N of bba's composing the bba on instance's class.

For each object, we build N bba's and combine them conjunctively. The resulting bba is the bba describing our belief about the value of the actual class to witch the object belongs.

Each bba has almost 2 focal elements:

1. The first is the actual class C of the object with bbm , m(C)= 1-P (P is a probability generated randomly).
2. The second is a subset θ of Θ (generated randomly) such that the actual class of the object under consideration belongs to θ and every of the other class belongs to θ with probability P. m(θ)= P

A larger $\mathbf{P}$ gives a larger degree of uncertainty.

4.3.3 Evaluation criteria

To evaluate a pruning method, we will consider the following parameters.

1. **The size** characterized by the number of nodes and leaves in the belief decision tree. It is generally accepted that the fewer terms in a model the better (Occam's razor).
2. **Accuracy** the accuracy of a classification method is determined by measuring the number of instances it, correctly, classifies among the total number of testing instances presented to the classifier. We will use two performance indicators are the **Percent of Correct Classification** (PCC) and **distance criterion** (dist_crit)(Elouedi et al., 2002a)

The reduction in size has usually been of secondary concern relative to classification accuracy. The improvement of the classification accuracy is a relevant criterion to judge the performance of the pruning method. Let us give a definition of these performance indicators:

Percent of Correct classification: PCC

The PCC represents the percent of the correct classification of the testing instances which are classified according to the induced decision tree. It is given by:

$$PCC = \frac{number\,of\,well\,classified\,instances}{total\,number\,of\,classified\,instances} * 100 \tag{4.1}$$

The PCC is computed as follows: for each testing instance, we make comparison between its real class (its class in the testing set) and the class given by the decision tree. Hence, the number of well classified instances represents the number of testing instances for which the class obtained by the tree is the same as their real class. In the framework of belief decision trees, we determine its corresponding leaf. Once it is found, we look for the most probable class corresponding to this leaf using the pignistic probability computed from the leaf's bba. The obtained class is considered as the class of the testing instance.

A PCC equal to 100% qualifies the belief decision tree as an excellent classifier, whereas a 'null' classifier has a PCC to 0%.

An equivalent criterion, also used in the literature, measure the proposition of incorrectly classified instances. This is knows as the *error rate (r=1-PCC).*

Distance criterion: ***dist_crit***

Distance criterion (Elouedi et al., 2002a) allowing to take account about all the beliefs characterizing the leaf's bba by comparing the pignistic probability induced from the testing instance's bba and its real class.

The distance criterion for a testing instance I_j belonging to a leaf L (its bba is $m^{\Theta}\{L\}$) is defined as follows:

$$dist\_crit\,(I_j) \;=\; Distance\,(BetP^{\Theta}\{L\}, C(I_j))$$

$$= \sum_{i=1}^{n} (BetP^{\Theta}\{L\}(C_i) - \delta_{j,i})^2 \tag{4.2}$$

Where the real class of the testing instance I_j is C(I_j), and $\delta_{j,i}$=1 if C(I_j)=C_i and 0 otherwise.

This distance verifies the following property.

$$0 \preceq dist\_crit(I_j) \preceq 2 \tag{4.3}$$

Next, we have to compute the average total distance relative to all the classified testing instances denoted dist_crit. So, we get:

$$dist\_crit = \frac{\sum_{I_j\, in\, classified\, instances} dist\_crit(I_j)}{total\, number\, of\, classified\, instances} \tag{4.4}$$

4.3.4 Cross validation

The construction of a belief decision tree requires a growing set for building it. Our pruning method requires pruning set for pruning it. We need a testing set for evaluating the performance of BDT before and after pruning.

The data set is divided into 10 parts. Nine parts are used as the training set, the last is used as the testing set. The procedure is repeated ten times, each time another part as the testing set. This method permits an unbaised estimation of the PCC. The training set is divided into 10 parts. Nine parts are used as growing set to build the tree, the last is used as the pruning set to select the best pruned tree.

4.3.5 Simulations on real databases

Description of the databases

As mentioned in the beginning of this section, we have performed simulations on real databases obtained from the UCI repository. A brief description

of these nominal-valued databases is presented in table # Data, # Gr,# Pr, #Ts, #attributes, #classes denote respectively the total number of instances in the database, the number of growing instances,the number of pruning instances, the number of testing instances, the number of attributes and the number of classes.

Table 4.1: Description of databases

Database	#Data	#Gr	#Pr	#Ts	#attributes	#classes
W. Breast Cancer	690	559	62	69	8	2
Balance Scale Weight	620	503	55	62	4	3
Congressional Voting	490	497	44	49	16	2

Experimental results

Let us remind that our objective is to reduce the size and improve the classification accuracy of belief decision tree by pruning it. In our simulation, we were interested in comparing the size, the PCC and the dist_crit of BDT in averaging and conjunctive approaches without pruning (Elouedi et al., 2001), with pre-pruning (Elouedi et al., 2002b) and with applying our pruning methods.

Our simulations will be performed for two cases, namely, the certain case and the uncertain case.

1. **The certain case :** The first case tests the efficiency of the pruning method in Quinlan algorithm when there is no uncertainty in classes by applying the averaging approach.

2. **The uncertain case :** The second case tests the efficiency of our pruning methods in both averaging and conjunctive approaches with many level of uncertainty (low, middle and high) according to the value of probability P.

Results of the certain case:

Tables 4.2, 4.3 and 4.4 summarize the different results relative to Wisconsin breast cancer, Balance Scale weight and Congressional voting databases

for the certain case. So, we compare the size, the PCC and the dist_crit of BDT without pruning (Size.bef.Prun, PCC.bef.Prun, dist_crit.bef.Prun), with pre-pruning (Size.aft.Pr.Prun, PCC.aft.Pr.Prun, dist.aft.Pr.Prun) and with applying our post-pruning method in averaging approach (Size.aft.Pt.Prun, PCC.aft.Pt.Prun, dist_crit.aft.Pt.Prun).

Table 4.2: Experimental measures (certain case (Size))

Degree of uncertainty	Size bef.Prun	Size aft.Pr.Prun	Size aft.Pt.Prun
W. Breast Cancer	274	151	123
B. Scale weight	326	265	109
C. Voting	57	34	29

Table 4.3: Experimental measures (certain case (PCC))

Degree of uncertainty	PCC bef.Prun	PCC aft.Pr.Prun	PCC aft.Pt.Prun
W. Breast Cancer	76.6%	77.2%	82.53%
B. Scale weight	60.7%	62.3%	70.9%
C. Voting	95.21%	95.74%	96.53%

Table 4.4: Experimental measures (certain case (dist_crit))

Degree of uncertainty	dist_crit bef.Prun	dist_crit aft.Pr.Prun	dist_crit aft.Pt.Prun
W. Breast Cancer	0.35	0.33	0.29
B. Scale weight	0.44	0.48	0.53
C. Voting	0.24	0.22	0.21

From these tables, we can conclude that our pruning method in certain case has good results. There are an improvement of the size of the tree in all databases. For Wisconsin Breast Cancer, the size goes from 274 items to 123 items. For Balance database, the size of the induced tree goes from 326 items to 109 items. Finally, for Congressional voting, the size is reduced from 57 to 29 items.

There are also an increase of the PCC for all databases. For Wisconsin Breast Cancer, the PCC goes from 76.6% to 82.53%. For Balance Scale

weight database, the PCC is increased from 60.7% to 70.9%. Finally, for congressional voting, the PCC goes from 95.21% to 96.53%.

There are an improvement of distance criterion but not for all databases. For Wisconsin Breast Cancer, the dist_crit goes from 0.35 to 0.29. For Balance Scale weight database, the dist_crit is increased from 0.44 to 0.53. Finally, for Congressional voting, the dist_crit is improved from 0.24 to 0.21.

From these tables, we can also conclude that pre-pruning reduces size, increases PCC and decreases distance criterion for all databases, but not better than our post-pruning method. For example in Wisconsin Breast Cancer database, the size goes from 274 items to 151 items, the PCC is increased from 76.6% to 77.2% and the distance is improved from 0.35 to 0.33.

Results of the uncertain case:

This section presents different results carried out from testing our pruning belief decision tree methods in averaging and conjunctive approaches on uncertain case.

Size

Tables 4.5, 4.6 and 4.7 compare the size of BDT before pruning (Size.bef. Prun_ave, Size.bef.Prun_conj) with the size after pre-pruning (Size.aft.Pr.Prun_ave, Size.aft.Pr.Prun_conj) and the size after our post-pruning methods (Size.aft. Pt.Prun_ave, Size.aft.Pt.Prun_conj) in both approaches.

Table 4.5: Experimental measures (W. breast cancer, uncertain case (Size))

Degree of uncertainty	Size.bef Prun_ave	Size.bef Prun_conj	Size.aft Pre.Prun_ave	Size.aft Pr.Prun_conj	Size.aft Pt.Prun_ave	Size.aft Pt.Prun_conj
low degree	399	352	302	292	82	81
middle degree	401	357	305	285	87	90
high degree	444	425	321	265	101	125
Mean	414	378	309	280	90	98

Table 4.6: Experimental measures (Balance Scale weight, uncertain case(Size))

Degree of uncertainty	Size.bef Prun_ave	Size.bef Prun_conj	Size.aft Pr.Prun_ave	Size.aft Pr.Prun_conj	Size.aft Pt.Prun_ave	Size.aft Pt.Prun_conj
low degree	508	489	338	323	139	125
middle degree	498	456	331	308	147	138
high degree	522	502	373	325	97	109
Mean	509	482	347	318	127	124

Table 4.7: Experimental measures (Congressional voting, uncertain case(Size))

Degree of uncertainty	Size.bef Prun_ave	Size.bef Prun_conj	Size.aft Pr.Prun_ave	Size.aft Pr.Prun_conj	Size.aft Pt.Prun_ave	Size.aft Pt.Prun_conj
low degree	171	202	133	142	75	70
middle degree	158	173	124	131	56	58
high degree	157	171	111	125	58	67
Mean	162	182	122	132	63	65

From Table 4.5, pruning belief decision tree methods in both approaches work well for all degree of uncertainty for Wisconsin Breast Cancer database. They lead to a reduction in size better than pre-pruning.

In averaging approach, the mean size goes from 414 items to 309 items with pre-pruning and to 90 items with our pruning method. In conjunctive approach, the mean size goes from 378 items to 280 items with pre-pruning and to 90 items with our pruning method.

For Balance Scale weight database, is the same thing. The size is reduced from 509 items to 347 with pre-pruning and reduced to 127 items with our pruning method in averaging approach. In conjunctive approach, The size is reduced from 482 items to 318 with pre-pruning and reduced to 124 items with our pruning method in averaging approach.

For Congressional voting database, the Table 4.7 shows that our pruning

belief decision tree methods have good results on belief decision tree. The size is improved from 162 items to 122 items with pre-pruning and improved to 63 items in averaging approach. The size is improved from 182 items to 132 items with pre-pruning and improved to 65 items in conjunctive approach.

From these tables, we can also conclude that the size after our post-pruning methods in both approaches is almost the same for all databases. For example, in Congressional voting database, there are 63 items in averaging approach and 65 items in conjunctive approach .

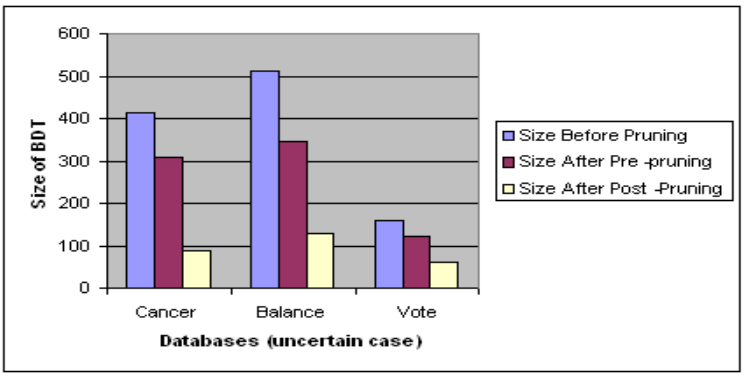

Figure 4.1: The size of BDT in averaging approach

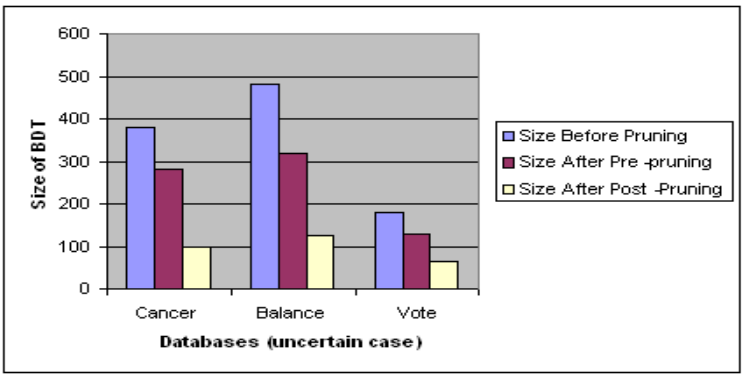

Figure 4.2: The size of BDT in conjunctive approach

PCC

Tables 4.8, 4.9 and 4.10 compare the PCC of BDT before pruning (PCC.bef. Prun_ave, PCC.bef.Prun_conj) with the PCC after pre-pruning (PCC.aft.Pr. Prun_ave, PCC.aft.Pr.Prun_conj) and the PCC after our post-pruning methods (PCC.aft.Pt.Prun_ave, PCC.aft.Pt.Prun_conj) in both approaches.

Table 4.8: Experimental measures (W. breast cancer, uncertain case(PCC))

Degree of uncertainty	PCC.bef Prun_ave	PCC.bef Prun_conj	PCC.aft Pr.Prun_ave	PCC.aft Pr.Prun_conj	PCC.aft Pt.Prun_ave	PCC.aft Pt.Prun_conj
low degree	67.97%	68.81%	69.13%	71.13%	82.38%	82.46%
middle degree	67.83%	68.18%	68.38%	70.76%	82.54%	81.01%
high degree	66.09%	68.1%	67.18%	71.03%	84.29%	82.17%
Mean	67.29%	68.36%	68.23%	70.97%	83.07%	81.88%

Table 4.9: Experimental measures (Balance Scale weight, uncertain case(PCC))

Degree of uncertainty	PCC.bef Prun_ave	PCC.bef Prun_conj	PCC.aft Pr.Prun_ave	PCC.aft Pr.Prun_conj	PCC.aft Pt.Prun_ave	PCC.aft Pt.Prun_conj
low degree	58.68%	63.41%	66.84%	67.26%	77.81%	76.15%
middle degree	58.35%	63.36%	66.58%	68.01%	77.78%	75.83%
high degree	63.1%	64.23%	67.26%	65.84%	81.7%	77.66%
Mean	60%	63.66%	66.89%	67.03%	79%	76.54%

Table 4.10: Experimental measures (Congressional voting, uncertain case(PCC))

Degree of uncertainty	PCC.bef Prun_ave	PCC.bef Prun_conj	PCC.aft Pr.Prun_ave	PCC.aft Pr.Prun_conj	PCC.aft Pt.Prun_ave	PCC.aft Pt.Prun_conj
low degree	94.29%	95%	94.88%	95.78%	95.78%	95.28%
middle degree	94.08%	94.88%	94.29%	95.21%	96%	95.76%
high degree	92.27%	92.17%	92.65%	93.02%	95.33%	95.71%
Mean	93.67%	94.01%	93.94%	94.43%	95.7%	95.58%

From Table 4.8, pruning belief decision tree methods in both approaches work well for all degree of uncertainty for Wisconsin Breast Cancer database. They lead to an increase in PCC better than pre-pruning.

In averaging approach, the mean PCC goes from 67.29% to 68.23% with pre-pruning and to 83.07% with our pruning method. In conjunctive approach, the mean PCC goes from 68.36% to 70.97% with pre-pruning and to 81.88% with our pruning method.

For Balance Scale weight database, is the same thing. The PCC is increased from 60% to 66.89% with pre-pruning and increased to 79% with our pruning method in averaging approach. In conjunctive approach, the PCC is increased from 63.66% to 67.03% with pre-pruning and increased to 76.54% with our pruning method in averaging approach.

For Congressional voting database, the Table 4.10 shows that our pruning belief decision tree methods have good results on belief decision tree. The PCC is improved from 93.67% to 93.94% with pre-pruning and improved to 95.7% with our pruning method in averaging approach. The PCC goes from 94.01% to 94.43% with pre-pruning and goes to 95.58% with our pruning method in conjunctive approach.

From these tables, we can also conclude that in most cases the mean PCC in averaging appraoch is better than in conjunctive approach. For example, in Balance Scale weight database, the PCC is 79% in averaging approach and 76.54% in conjunctive appraoch. In Congressional voting database, the PCC is the same in both appraoches.

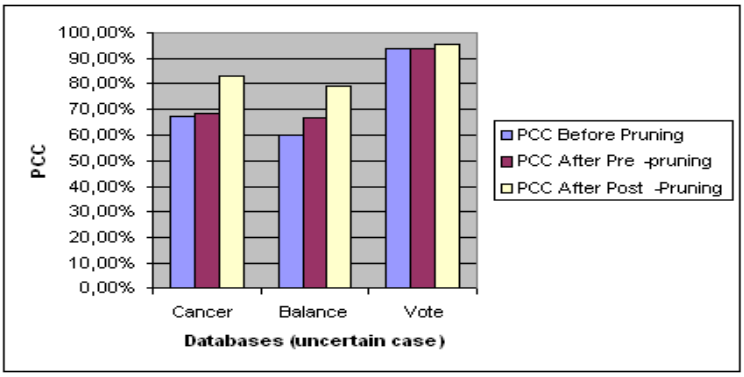

Figure 4.3: The PCC of BDT in averaging approach

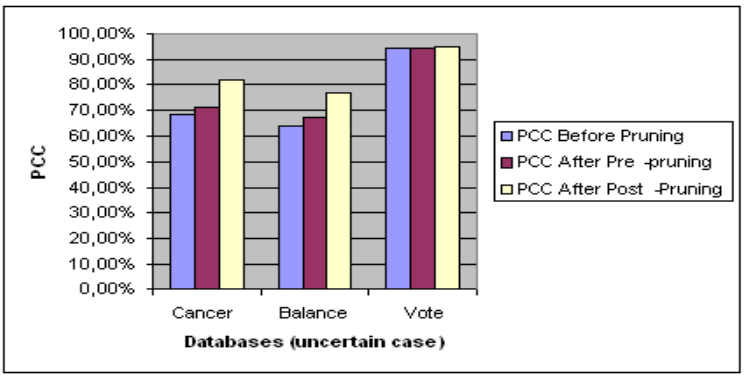

Figure 4.4: The PCC of BDT in conjunctive approach

Distance criterion: dist_crit

Tables 4.11, 4.12 and 4.13 Compare the distance of BDT before pruning (dist.bef.Prun_ave, dist.bef.Prun_conj) with the distance after pre-pruning (dist.aft.Pr.Prun_ave, dist.aft.Pr.Prun_conj) and the distance after our post-pruning methods (dist.aft.Pt.Prun_ave, dist.aft.Pt.Prun_conj) in both approaches.

Table 4.11: Experimental measures (W. breast cancer, uncertain case(dist))

Degree of uncertainty	dist.bef Prun_ave	dist.bef Prun_conj	dist.aft Pr.Prun_ave	Size.aft Pr.Prun_conj	dist.aft Pt.Prun_ave	dist.aft Pt.Prun_conj
low degree	0.42	0.4	0.37	0.34	0.34	0.32
middle degree	0.39	0.39	0.35	0.37	0.32	0.31
high degree	0.38	0.36	0.36	0.31	0.29	0.27
Mean	0.39	0.38	0.36	0.34	0.31	0.3

Table 4.12: Experimental measures (Balance Scale weight, uncertain case(dist))

Degree of uncertainty	dist.bef Prun_ave	dist.bef Prun_conj	dist.aft Pr.Prun_ave	dist.aft Pr.Prun_conj	dist.aft Pt.Prun_ave	dist.aft Pt.Prun_conj
low degree	0.45	0.42	0.43	0.42	0.41	0.4
middle degree	0.39	0.4	0.38	0.39	0.38	0.37
high degree	0.49	0.47	0.46	0.44	0.39	0.41
Mean	0.44	0.43	0.42	0.41	0.39	0.39

Table 4.13: Experimental measures (Congressional voting, uncertain case(dist))

Degree of uncertainty	dist.bef Prun_ave	dist.bef Prun_conj	dist.aft Pr.Prun_ave	dist.aft Pr.Prun_conj	dist.aft Pt.Prun_ave	dist.aft Pt.Prun_conj
low degree	0.24	0.25	0.23	0.21	0.22	0.19
middle degree	0.26	0.22	0.25	0.24	0.26	0.21
high degree	0.2	0.19	0.22	0.17	0.21	0.18
Mean	0.23	0.22	0.23	0.2	0.23	0.19

From Table 4.11, pruning belief decision tree methods in both approaches work well for all degree of uncertainty for Wisconsin Breast Cancer database. They lead to a reduction in distance better than pre-pruning.

In averaging approach, the mean distance goes from 0.39 to 0.36 with pre-pruning and to 0.31 with our pruning method. In conjunctive approach, the mean distance goes from 0.38 to 0.34 with pre-pruning and to 0.3 with our pruning method.

For Balance Scale weight database, is the same thing. The distance is reduced from 0.44 to 0.42 with pre-pruning and to 0.39 with our pruning method in averaging approach. In conjunctive approach, The distance is improved from 0.43 to 0.41 with pre-pruning and to 0.39 with our pruning method.

For Congressional voting database, the Table 4.13 shows that our pruning belief decision tree methods have good results on belief decision tree. The distance keeps the same value 0.23 with pre-pruning and with our pruning method in averaging approach. The distance is improved from 0.22 to 0.2 with pre-pruning and to 0.19 with our pruning method in conjunctive approach.

From these tables, we can also conclude that the distance after our pruning method in conjunctive approach is better than the distance after our pruning method in averaging approach for all databases. For example, in Congressional voting database, the distance is 0.23 in averaging approach and 0.19 in conjunctive approach.

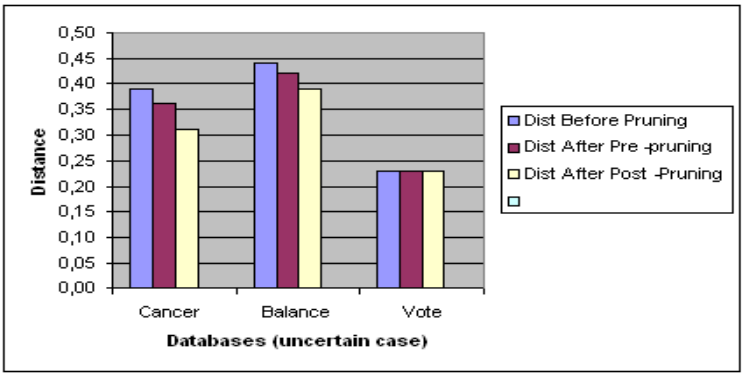

Figure 4.5: The distance of BDT in averaging approach

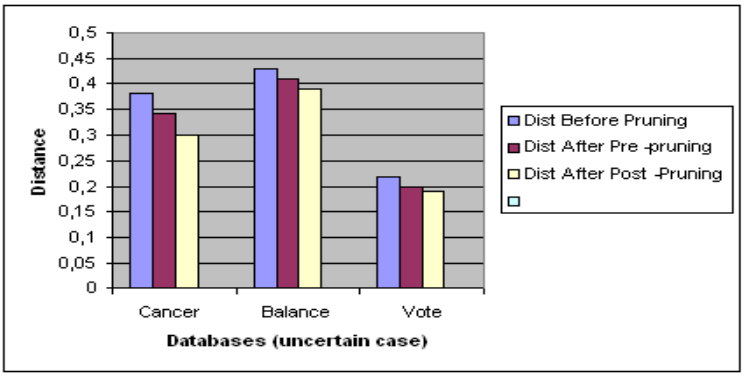

Figure 4.6: The distance of BDT in conjunctive approach

4.4 Conclusion

In this chapter, we have shown the different results obtained from simulations and that have been performed on real databases in both certain and uncertain cases. These experimentations have shown interesting results for both cases with the objective to reduce the size of the induced tree and improve the classification accuracy.

Conclusion

Inducing a belief decision tree (BDT) from real world databases with averaging or conjunctive approaches, may lead in most cases to very large trees with bad classification accuracy due to uncertainty. The results are many of undesirable nodes and difficulty to interpret the tree. The pruning copes with this problem of overfitting the data, improves the comprehensibility of decision tree and also increases the quality of classification.

In this master thesis, we have defined post-pruning belief decision tree methods based on standard minimal cost-complexity pruning method to prune the belief decision tree in averaging and conjunctive approaches with the objective to reduce its size and to improve its classification accuracy.

Our pruning belief decision tree method in averaging approach is close to the belief decision tree building procedure in averaging approach, which it is an extension of Quinlan method. Where as in conjunctive approach, our pruning method is more close to the basics of the Transferable Belief Model (TBM), one interpretation of belief function theory, like in its building phase.

In our work, we have performed simulations on real databases in order to evaluate the performance of our pruning belief decision tree methods comparing with BDT without pruning and with pre-pruning. Two evaluation criteria are taken into account namely the size of the tree and its classification accuracy.

Results of experimentations on a certain case show that our pruning methods perform well for all databases. There are a reduction of size, and improvement of classification accuracy (Percent of correct classification and distance criterion) of BDT.

Results on an uncertain case show that our pruning method has a good impact on the size and the classification accuracy of the belief decision tree for different degree of uncertainty and for all databases in both averaging and conjunctive approaches.

Finally, regarding the interesting results obtained in this work, we could propose further works that may be done to improve our pruning method. So, we can develop other pruning BDT methods based on other standard pruning methods. It will be also interesting to extend the belief decision tree approach to handle continuous attributes and the uncertainty in attributes values in the training set. We can also apply the techniques of boosting and bagging to the BDT in order to improve its complexity and accuracy.

Appendix A

Implementation

A.1 Introduction

In order to test our two methods in averaging and conjunctive approaches, we have developed programs in Matlab V6.5. Obviously, we have implemented the pruning algorithm as a well as selection algorithm that we have detailed in Chapter 3. The outputs of our programs are basically:

1. A series of pruned decision trees induced form the two methods (averaging and conjunctive)
2. The best pruned tree has the best Percent of Correct Classification, denoted PCC on testing set.

A.2 Principal variables

In this section, we present the list of the major variables that are used in our developed algorithms (it is not an exhaustive list):

- *training_set* includes the attributes values of all the training instances, the beliefs on their classes and also their real classes.
- *number_attributes* is the number of attributes relative to the given classification problem.
- *number_classes* is the number of classes relative to the given classification problem.

- *node*(either a leaf or decision node) each node is composed of:
  - *.type* which maybe a leaf or an attribute.
  - *.level* is the level of the node in belief decision tree. It is equal to 0 for the root node.
  - *.name* is equal to 0 when it is an attribute, otherwise (when it is an attribute)contains the name of the attribute (its position).
  - *.beliefs* is computed as an average bba or as a joint bba of the instances belonging to the node (according to the applied approach).
  - *.gr* is equal to 0 when it is a leaf, otherwise is the gain ratio or the diff ratio of the node.
  - *.pignistic* is computed as the average pignistic of the instances belonging to the node (needed in the case of pruning belief decision tree method of averaging approach
  - *.nbr* represents the number of nodes.
  - *.path_attributes* represents the attributes leading to this node.
  - *.path_values* represents the values of the attributes leading to this node.
  - *.path_nodes* represents the nodes leading to this node.
  - *.pred* represents the parent of the node.
  - *.nbsucc* is equal to 0 when it is a leaf, otherwise (when it is an attribute), it contains item the number of possible values of this attribute.
- *tree_averaging* is the belief decision tree induced from the application of the averaging approach. It is composed by nodes(decision nodes and leaves).
- *tree_conjunctive* is the belief decision tree induced from the application of the conjunctive approach. It is composed by nodes(decision nodes and leaves)
- *series_pruned_trees* is increasingly pruned trees.
- *error_rate_node* is the error rate if the node is pruned. It becomes a leaf.

- *distance_node* is the distance between objects if the node is pruned. It becomes a leaf.
- *error_rate_subtree* is the error rate if the node is not pruned. It stills a subtree.
- *distance_subtree* is the distance if the node is not pruned. It stills a subtree.
- *alpha_ave* is the error rate per pruned leaf.
- *alpha_conj* is the distance per pruned leaf.
- *number_leaves* is the number of leaves in the subtree.
- *best_tree* is the best pruned tree from the series_pruned_trees
- *vector_pignistic* is the pignistic of the belief on the classes of the instances at this node.
- *pruned_tree* is a tree after a phase of pruning.
- *testing_set* includes the attribute values of all the testing instances, the beliefs on their classes and also their real classes that will be compared with the classes given by the induced belief decision tree,used to select the best tree.
- *PCC* is the percent of the correct classification of the instances to classify.
- *Distance* is the value of distance criterion of the instances to classify.

A.3 Principal Programs

We present the different programs that we have developed in order to prune the belief decision trees using averaging and conjunctive approaches.

Pruning phase

The pruning phase is considered as important phase, as in the averaging approach and in the conjunctive one. It consists in generating a series of

increasingly pruned trees.

Averaging approach

- **Pruning_averaging** is the principal program to prune the belief decision tree according to the averaging approach.
- **Error_pruning:** is a function computing the error rate if the node is pruned.
- **Error_non_pruning:** is a function computing the error rate if the node is not pruned.
- **Compute_alpha_ave:** is a function computing the alpha of a node.
- **Prune_node:** is a function pruning a node from a tree.

Conjunctive approach

- **Pruning_conjunctive** is the principal program to prune the belief decision tree according to the conjunctive approach.
- **Distance_pruning:** is a function computing the distance if the node is pruned.
- **Distance_non_pruning:** is a function computing the distance if the node is not pruned.
- **Compute_alpha_conj:** is a function computing the alpha of a node.
- **Prune_node:** is like averaging approach.

Selection phase

This set of procedures ensures the selection of the best pruned belief decision tree. It has the best classification accuracy on testing set. These procedures are the same in the two approaches.

- **Best_pruned_tree:** is a function allowing to identify the best pruned belief decision tree. it has the best PCC on testing set.

- **Classification_results:** is a function computing the value of classification results on a testing set.
- **Classify:** is a function give the class of an instance.

A.4 Pruning belief decision tree algorithms

In this section, we present the major algorithms relatives to pruning belief decision tree methods in averaging and conjunctive approaches. Two phases will be considered pruning phase and selection phase.

Pruning Phase

Averaging approach

Algorithm Pruning_averaging
Input: tree_averaging, number_attributes, number_classes, number_training_instances
Output: series_pruned_trees

1. **begin**
2. (* Initialization *)
3. number_nodes ← 0;
4. series_pruned_trees(1,:) ← tree_averaging;
5. number_pruned_trees ← 1;
6.
7. (* Stop pruning if only the root is left *)
8. **While** number_nodes ≠ 1
9. **do** number_node_treated ← 0;
10. vector_alpha ← [];
11.
12. (* For each node computing alpha *)
13. **for** i ← 2 **to** length (tree_averaging)
14. **do if** isequal (tree_averaging(i).type, 'attribute')
15. (* Computing the error rate if the node is pruned and not pruned *)

```
16.                 then error_rate_node ←Error_pruning(tree_averaging, i,
                         number_training_instances)
17.                     [error_rate_subtree,number_leaves] ← Error_non_pruning
                        (tree_averaging, i, number_training_instances)
18.                     (* Computing the error rate per pruned leaf, denoted
                        alpha *)
19.                     alpha ← Compute_alpha (error_rate_node, error_rate_subtree,
                        number_leaves);
20.                     number_node_treated ← number_node_treated+1;
21.                     vector_alpha (number_node_treated).alpha ← alpha;
22.                     vector_alpha (number_node_treated).nbr ← tree_averaging(i).nbr;
23.               end if
24.          end for
25.
26.          (* Finding the node has the minimum alpha and pruning it *)
27.          min_alpha ← Find_min_alpha (vector_alpha)
28.          pruned_tree ← Prune_node(tree_averaging, min_alpha);
29.          number_pruned_trees ← number_pruned_trees +1;
30.          for j ← 1 to length(pruned_tree)
31.             do series_pruned_trees(number_pruned_trees, j) ← tree_averaging(j);
32.          end for
33.          tree_averaging ← series_pruned_trees(number_pruned_trees, :);
34.
35.          (* Computing number of nodes in the pruned tree *)
36.          number_nodes ← 0;
37.          for i ← 1 to length(tree_averaging);
38.             do if isequal(tree_averaging(i).type,'attribute')
39.                     then number_nodes ← number_nodes+1;
40.             end if
41.          end for
42. end while
43. end
```

Algorithm Error_pruning
Input: tree_averaging, indice_node, number_training_instances
Output: error_rate_node

1. **begin**
2. (*Finding the pignistic of the belief on the classes of the instances at this node*)
3. vector_pignistic ← tree_averaging(indice_node).pignistic;
4. (*Finding the class has the max pignistic *)
5. max_pignistic ← **Maxim**(vector_pignistic);
6. indice_max ← **Index_max**(vector_pignistic, max_pignistic);
7. vector_error ← vector_pignistic;
8. vector_error(:, indice_max(1))←[];
9. (* Computing the error rate *)
10. sum_error ← sum(vector_error);
11. error_rate_node ← sum_error/ number_training_instances;
12. **end**

Algorithm Error_non_pruning
Input: tree_averaging, indice_node, number_training_instances
Output: Error_rate_subtree, number_leaves

1. **begin**
2. number_leaves ← 0;
3. error_rate_subtree ← 0;
4. **for** j ← 1 **to** length(tree_averaging)
5. **do if** isequal(tree_averaging(j).type,'leaf')
6. **then** path ← tree_averaging (j).path_nodes;
7. exist ← 0;
8. **for** t← 1 **to**length(path)
9. **do if** path(t)← tree_averaging (indice_node).nbr;
10. **then** exist ← 1;
11. **end if**
12. **end for**
13. **if** exist=1
14. **then** number_leaves ← number_leaves +1;
15. vector_pignistic ← tree_averaging(j).pignistic;
16. (*Finding the max class)
17. max ← **Maxim**(vector_pignistic);

```
18.                                indice_max ← Index_max(vector_pignistic,max);
19.                                vector_error ← vector_pignistic;
20.                                vector_error(:,indice_max(1))← [];
21.                                (*Computing the error rate*)
22.                                sum_error ← sum(vector_error);
23.                                error_rate_subtree ← error_rate_subtree+(sum_error/
24.                                number_training_instances);
25.                        end if
26.          end if
27.  end for
28.  end
```

Algorithm Compute_alpha_ave
Input: error_rate_node, error_rate_subtree, number_leaves
Output: alpha

```
1.  begin
2.  alpha ←( error_rate_node - error_rate_subtree)/( number_leaves -1);
3.  end
```

Algorithm Prune_node
Input: tree_averaging, min_alpha
Output: pruned_tree

```
1.  begin
2.  number_items ← 0;
3.  (*The node become a leaf*)
4.  for i← 1 to length (tree_averaging)
5.          do if tree_averaging(i).nbr==min_alpha.nbr
6.                  then tree_averaging(i).type ← 'leaf';
7.                          tree_averaging(i).nbr ← [];
8.                  end if
9.          exist ← 0;
10.         for t ← 1 to length(tree_averaging(i).path_nodes);
11.                 do if tree_averaging(i).path_nodes(t)== min_alpha.nbr
12.                         then exist ← 1;
13.                 end if
```

```
14.       end for
15.       if exist=0
16.             then number_items ← number_items +1;
17.                   pruned_tree(number_items) ← tree_averaging(i);
18.       end if
19. end for
20. end
```

Conjunctive approach

Algorithm Pruning_conjunctive
Input: tree_conjunctive, number_attributes, number_classes, number_training_instances
Output: series_pruned_trees

```
1.  begin
2.  (* Initialization *)
3.  number_nodes ← 0;
4.  series_pruned_trees(1,:) ← tree_conjunctive;
5.  number_pruned_trees ← 1;
6.
7.  (* Stop pruning if only the root is left *)
8.  While number_nodes ≠ 1
9.        do number_node_treated ← 0;
10.          vector_alpha ← [];
11.
12.          (* For each node computing alpha_conj *)
13.           for i ← 2 to length (tree_conjunctive)
14.               do if isequal (tree_conjunctive(i).type, 'attribute')
15.                        (* Computing the distance if the node is pruned and
                             not pruned *)
16.                    then distance_node ←Distance_pruning(tree_conjunctive,
    i,
                             number_training_instances)
17.                        [distance_subtree,number_leaves] ← Distance_non_pruning
                           (tree_conjunctive, i, number_training_instances)
```

```
18.                     (* Computing the distance per pruned leaf, denoted
                        alpha_conj *)
19.                     alpha_conj ← Compute_alpha_conj (distance_node, dis-
    tance_subtree,                  number_leaves);
20.                     number_node_treated ← number_node_treated+1;
21.                     vector_alpha (number_node_treated).alpha ← alpha;
22.                     vector_alpha (number_node_treated).nbr ← tree_conjunctive(i).nbr;
23.                 end if
24.             end for
25.
26.             (* Finding the node has the minimum alpha and pruning it *)
27.             min_alpha ← Find_min_alpha (vector_alpha)
28.             pruned_tree ← Prune_node(tree_conjunctive, min_alpha);
29.             number_pruned_trees ← number_pruned_trees +1;
30.             for j ← 1 to length(pruned_tree)
31.                 do series_pruned_trees(number_pruned_trees, j) ← tree_conjunctive(j);
32.             end for
33.             tree_conjunctive ← series_pruned_trees(number_pruned_trees, :);
34.
35.             (* Computing number of nodes in the pruned tree *)
36.             number_nodes ← 0;
37.             for i ← 1 to length(tree_conjunctive);
38.                 do if isequal(tree_conjunctive(i).type,'attribute')
39.                         then number_nodes ← number_nodes+1;
40.                 end if
41.             end for
42.     end while
43.     end
```

Algorithm Distance_pruning
Input: tree_conjunctive, indice_node, number_training_instances
Output: distance_node

```
1. begin
2. (*Finding the beliefs on the classes of the instances at this node*)
3. matrice_beliefs ← tree_conjuctive(indice_node).beliefs;
```

```
4. (* Computing the distance *)
5. distance_node ← SumD(matrice_beliefs)/number_training_instances;
6. end
```

Algorithm Distance_non_pruning
Input: tree_conjunctive, indice_node, number_training_instances
Output: distance_subtree, number_leaves

```
1. begin
2. number_leaves ← 0;
3. distance_subtree ← 0;
4. for j ← 1 to length(tree_conjunctive)
5.         do if isequal(tree_conjunctive(j).type,'leaf')
6.                 then path ← tree_conjunctive (j).path_nodes;
7.                         exist ← 0;
8.                         for t← 1 tolength(path)
9.                            do if path(t)← tree_conjunctive(indice_node).nbr;
10.                                 then exist ← 1;
11.                           end if
12.                        end for
13.                        if exist=1
14.                           then number_leaves ← number_leaves +1;
15.                                matrice_beliefs ← tree_conjunctive(j).beliefs;
16.                                (*Computing the distance in the subtree*)
17.                                sum_dist ← SumD(matrice_beliefs);
18.                                distance_subtree ← distance_subtree+(sum_dist/
19.                                number_training_instances);
20.                        end if
21.        end if
22. end for
23. end
```

Algorithm Compute_alpha_conj
Input: distance_node, distance_subtree, number_leaves
Output: alpha_conj

```
1. begin
```

2. alpha_conj ←(distance_node - distance_subtree)/(number_leaves -1);
3. **end**

Selection phase

Algorithm Best_pruned_tree
Input: series_pruned_trees, testing_set, number_ckasses, number_attributes
Output: best_tree

```
1.      begin
2.      (* Computing the PCC of each pruned tree *)
3.      for i ← 1 to length(series_pruned_trees)
4.         do pruned_tree ← series_pruned_trees (i,:);
5.            [pcc,kappa,distance] ← Classification_results(testing_set, pruned_tree);
6.            vector_pcc(i).nbr ← i;
7.            vector_pcc(i).pcc ← pcc;
8.      end for
9.
10.     (*Finding the best tree has the best PCC *)
11.     best_pcc ← max(1)
12.     for i ← 2 to length(series_pruned_trees)
13.     do if vector_pcc(i).pcc ≥ best_pcc.pcc
14.           then best_pcc ← vector_pcc(i);
15.     end if
16.     end for
```

17. (* Finding the best tree *)
18. best_tree ← series_pruned_trees (best_pcc.nbr,:)
19. **end**

Algorithm Classification_results
Input: testing_set, number_calsses, number_attributes, pruned_tree
Output: PCC, distance

1. **begin**
2. (* Initialization*)
3. [n_cases, nc] ← size(testing_set);
4. instances_to_classify ← **Getting_attribute_values**(testing_set,n_cases,number_attributes);

```
5. class_truth ← testing_set (:,nc);
6. truth_prediction_ave ← zeros(number_classes, number_classes +1);
7. dist_class ← zeros(n_cases,1);
8.
9. (* Treatment *)
10. for i ← 1 to n_cases
11.         do[beliefs,probab,class] ← classify(instances_to_classify(i,:),
12.           number_classes, pruned_tree);
13.           class_result_ave ← [class_result_ave; class];
14.           class_result_probab ←[class_result_probab; probab];
15.           truth_prediction_ave(class_truth(i,1),class_result_ave(i,1)) ← truth_
              prediction_ave(class_truth(i,1),class_result_ave(i,1)) + 1;
16.           if class_result_ave(i) ≼ number_classes
17.                then dist_class(i) ← sum(power(class_result_probab(i,:),2));
18.                     dist_class(i) ← dist_class(i)-power(class_result_probab(i,
                        class_truth(i)),2)+ power(1-class_result_probab(i,class_truth(i)),2);
19.           end if
20. end for
21.
22. (* Computing PCC *)
23. ttrace ← trace(truth_prediction_ave);
24. nb_classed_cases ← n_cases-sum(truth_prediction_ave(:,number_classes+1));
25. PCC ← ttrace/nb_classed_cases
26. (* Computing the distance criterion *)
27. distance ← sum(dist_class)/nb_classed_cases
28. end
```

A.5 Conclusion

In this appendix, we have presented the list of the major variables that are used in our developed programs and we have detailed the main algorithms relative to our pruning belief decision tree methods in both approaches.

Bibliography

Barnett, J. A. (1991). Calculating Dempster-Shafer plausibility. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 13(6), pp 599-602.

Ben Amor, N., Benferhat, S., & Elouedi, Z. (2004). Qualitative classification with possibilistic decision trees. In Proceedings of the International Conference on Information Processing of Uncertainty in Knowledge Based Systems. IPMU'2004. Perugia, Italia.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and Regression Trees. Wadsworth, Belmont, CA.

Dempster, A. P. (1967). Upper and lower probabilities induced by a multiple valued mapping. Annals of Mathematical Statistics, vol 38, pp 325-339.

Cestnik, B., & Bratko, I. (1991). On Estimating Probabilities in Tree Pruning. Machine Learning: EWSL-91, Y. Kodratoff, ed., Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag, no 482, pp 138-150.

Dempster, A. P. (1968). A generalisation of Bayesian inference. Journal of the Royal Statistical Society, Series B, vol 30, pp 205-247.

Denoeux, T. (1999). Reasoning with imprecise belief structures. International Journal Approximate Reasoning, vol 20, pp 70-111.

Denoeux, T., & Skarstien-Bajanger, M. (2000). Induction of decision trees for partially classified data. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (pp 2923-2928). Nashiville, USA:IEEE.

Elouedi, Z., Mellouli, K., & Smets, P. (2000). Decision trees using the belief function theory. In Proceedings of the international conference on Information

Processing and Management of Uncertainty IPMU'2000, vol1, pp 141-148, Madrid, Sapin, July 2000.

Elouedi, Z., Mellouli, K., & Smets, P. (2001). Belief decision trees: Theoretical foundations. In International Journal of Approxiamate Journal IJAR, vol 28(2-3), pp 91-124, 2001.

Elouedi, Z., Mellouli, K., & Smets, P. (2002a). Belief Decision Tree: A New Decision Tree Apprroach under Uncertainty. Thesis, Institut Supérieur de Gestion de Tunis, Tunisie, 2002.

Elouedi, Z., Mellouli, K., & Smets, P. (2002b). A Pre-Pruning Method in Belief Decision Trees. The Ninth International conference on Information Processing and Management of uncertainty in knowledge-Based Systems IPMU 2002, Vol 1, Annecy, France(2002), pp 579-586.

Esposito, F., Malerba, D., & Semeraro, G. (1997). A Comparative Analysis of Methods for Pruning Decision Trees. IEEE Pattern analysis and Machine Intelligence, vol 19(5), pp 476-491.

Hüllermeier, E. (2002). Possibilistic Induction in decision tree learning. In Proceedings of the 13 Th European Conference on Machine Learning. ECML2002. pp 173-184. Helsinki, Finland.

Jenhani, I., Elouedi, Z. Ben Amor, N. & Mellouli, K. (2005). Qualitative inference in possibilistic option decision trees. In Proceedings of the Eight European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty.

Kholas, J., & Monney, P. A. (1995). A mathematical theory of hints: An approach to Dempster-Shafer theory of evidence. Lecture Notes in Economics and Mathematical Systems 425, Springer-Verlag.

Marsala, C. (1998). Apprentissage inductif en prsence de donnes imprcises: Construction et utilisation d'arbres de decision flous. PhD thesis, Université Paris6, LIP6, France.

Mellouli, K. (1987). On the propagation of beliefs in networks using the Dempter-Shafer theory of evidence. PhD thesis, School of business, University of Kansas.

Mingers, J. (1987). Expert systems - rule induction with statistical data. Journal of the operational research society vol 38 pp 39-47,1987.

Mingers, J. (1989). An empirical Comparison of Pruning Methods for Decision Tree Induction. Machine Learning, vol 4, no 2, pp 227-243, 1989.

Niblett, T. & Bratko,I. (1986). Learning decision rules in noisy domains. Proc. Expert Systems 86, Cambridge: Cambridge University Press.

Quinlan, J.R. (1986). Induction of decision trees. Machine Learning, vol 1, pp 81-106.

Quinlan, J.R. (1987a). Decision trees as probabilistic classifiers. In Proceedings of the Fourth International on Machine Learning, pp 31-37. Morgan Kaufmann.

Quinlan, J.R. (1987b). Probabilistic decision tree. In R.S. Michalski, J.G. Carbonell, & T.M Michell (Eds.), Machine Learning, vol 3, Chap.5 (pp 267-301). Morgan Kaufmann.

Quinlan, J.R. (1987c). Simplifying decision trees. International Journal of Man-Machine studies, pp 221-234.

Quninlan, J.R., & Riverst, L.R. (1989). Inferring Decision Trees Using the Minimum description length Principle," Information and Computation, vol 80, pp 227-248, 1989.

Quinlan, J.R. (1990a). Decision trees and decision making. IEEE transactions on Systems, Man and Cybernatics, vol 20(2), pp 339-346.

Quinlan, J.R. (1993). C4.5: Programs for Machine learning. Morgan Kaufman, San Mateo, california.

Shafer, G. (1976). A matimatical theory of evidence. Princeton Univ. Press. Princeton, NJ.

Smets, P. (1988). Belief functions. In P. Smets, E. H. Mamdani, D. Dubois, & H.. Prade (Eds.), Non standard logics for automated reasoning, pp 253-286. Academic Press, London.

Smets, P., & Kennes, R. (1994). The transferable belief model. Artificial Intelligence, vol 66, pp 191-236.

Smets, P. (1995). The canonical decomposition of a weighted belief. In Proceedings of the 14th international joint conference on artificial intelligence (pp. 1896-1901). Montreal, Canada: IJCAI'95, Morgan Kaufmann.

Smets, P. (1998a). The transferable belief model for quantified belief representation. In D.M. Gabbay & P. Smets (Eds), Handbook of defeasible reasoning and uncertainty management systems, vol 1, pp 207-301. Doordrecht, The Netherlands: Kluwer.

Smets, P. (1998b). The application of belief transferable belief model to diagnostic problems. International Journal of Intelligent Systems, vol 13, pp 127-157.

Smets, P., & Kruse, R. (1997). The transferable belief model for belief representation. In A. Motro & P. Smets (Eds.), Uncertainty in information Systems: From needs to solutions, pp 343-368. Boston, MA: Kluwer.

Umano, M., Okamoto, H., Tamura, I. H. H., Kawachi, F., Umededzu, S., & Kinoshita, J. (1994). Fuzzy decision trees by fuzzy ID3 algorithm and its application to diagnosis systems. In Proceedings of the Third IEEE International Conference on Fuzzy Systems, vol 3, pp 2113-2118. Orlondo, USA: FUZZ- IEEE'94.

Utgoff, P. E. (1988). ID5: An incremental ID3. In Proceedings of the Fifth International Conference on Machine Learning, pp 107-120. Morgan Kaufman.

Utgoff, P. E. (1989). Incremental induction of decision trees. Machine Learning, vol 4, pp 161-186.

Walley, P. (1991). Statistical reasoning with imprecise probabilities. Chaman and Hall, London.

Yuan, Y., & Shaw, M. J. (1995). Induction of fuzzy decision trees. Fuzzy sets and systems, vol 69,pp 125-139.

Zeidler, J., & Schlosser, M. (1996). Continuous valued attributes in fuzzy decision trees. In Procedings of the sixth International Conference on Information Processing and Management of Uncertainty, vol1,pp 395-400. Spain:IPMU'96.