FORMALIZATION AND COMPARISON OF SPLIT CRITERIA FOR DECISION TREES

By
Laura Elena Răileanu

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTEUR ÈS SCIENCES
AT
UNIVERSITY OF NEUCHÂTEL
NEUCHÂTEL, SWITZERLAND
MAY 2002

UNIVERSITY OF NEUCHÂTEL
DEPARTMENT OF
COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Sciences for acceptance a thesis entitled "**Formalization and Comparison of Split Criteria for Decision Trees**" by **Laura Elena Răileanu** in partial fulfillment of the requirements for the degree of **Docteur ès sciences.**

Dated: May 2002

Research Supervisor: _____
Prof. Dr Kilian Stoffel, University of Neuchâtel

Examing Committee: _____
Prof. Dr Hans-Heinrich Nägeli, University of Neuchâtel

_____
Prof. Dr Boi Faltings, Swiss Federal Institute of Technology, Lausanne

UNIVERSITY OF NEUCHÂTEL

Date: **May 2002**

Author: **Laura Elena Răileanu**

Title: **Formalization and Comparison of Split Criteria for Decision Trees**

Department: **Computer Science**

Degree: **Ph.D.** Convocation: **May** Year: **2002**

To my parents.

Table of Contents

List of Figures

IMPRIMATUR POUR LA THESE

Formalization and comparison of split criteria for decision trees

de Mme Laura Elena Raileanu

UNIVERSITE DE NEUCHATEL

FACULTE DES SCIENCES

La Faculté des sciences de l'Université de Neuchâtel sur le rapport des membres du jury,

MM. K. Stoffel (directeur de thèse), H.-H. Naegeli et B. Faltings (Lausanne)

autorise l'impression de la présente thèse.

Neuchâtel, le 31 mai 2002

Le doyen:

F. Zwahlen

Abstract

The grand challenge of Knowledge Discovery in Databases (KDD) is to automatically process large quantities of raw data, identify the most significant and meaningful patterns, and present this knowledge in an appropriate form for achieving the user's goal. Knowledge discovery systems face challenging problems from the real databases that tend to be very large, redundant, noisy and dynamic.

The classification process (which finds a set of models or functions that describe and distinguish data classes or concepts) is done for the purpose of being able to use the model to predict the class of objects whose class label is unknown.

This thesis addresses a powerful tool, used in classification and prediction, namely *decision trees*, in the context of large data sets.

After a survey of relevant issues in decision trees, we concentrate our attention to decision split criteria, which have a crucial importance in the decision trees construction. Different goodness measures, evolving from different contexts (machine learning or statistics) were used in the selection of test attributes. It is not obvious which of the split criteria will produce the "best" decision trees for a given data set. We introduce a formal methodology, which allows us to analytically compare multiple split criteria. This permits us to present fundamental insights into the decision process. Furthermore, we are able to present a formal description of how to select between split criteria for a given data set. As an illustration we apply the methodology to the two widely used split criteria: Gini Index and Information Gain. This mathematical comparison conducts us to affirm that there are some differences in the choice of an attribute of split between the two criteria. Therefore, the structure of an decision tree depends on the decision split criterion. Numerically, the number of cases in which the two criteria are selecting differently is so small, that we can say that in general the split criteria choose similarly.

In the actual context of increasing sizes of databases, a natural concern is the behavior of these split functions when the size of the data set, from which the decision trees are constructed, increases. In this purpose, a new family of split functions, including the well

known ones, is introduced. The behavior of the whole family of split functions is evaluated with respect to the increasing sizes of databases, by taking into consideration the following criteria: tree size, error rate for the training data, error rate for the test data. The conducted tests have revealed that the size of the databases definitely influences the behavior of the split functions. Especially the widely used Information Gain is very sensitive to the size of the training set. The simplest split functions of the introduced family behave in a very stable way and, furthermore, the created trees are superior to the trees inferred using the Gini Index or the Information Gain based on our evaluation criteria. This is important, especially in a KDD context as the size of the training sets can vary from very small to very large.

When decision trees have to be constructed from very large data sets, a natural idea is to perform sampling. Our new technique of sampling data sets guarantees that the decision trees inferred from the whole database as well as from the sampled database are almost always identical. This affirmation is sustained by our formal analysis of the sampling technique that is followed by evaluations on real data sets.

The formal methodology introduced in this thesis combined with the new family of split functions we defined will furnish to the user the way to choose the appropriate criterion to construct decision trees from large data sets in function of its specific requirements like accuracy and simplicity of rules.

Acknowledgements

I would like to thank Professor Kilian Stoffel, my supervisor, for his helpful direction and constant support during this research. I owe him my gratitude for the encouragement given in the undertaking of this work.

I would like to thank all the members of my committee for their feedback and careful readings of the thesis.

Financial support for the research in this thesis was gratefully received, in part, from the Swiss National Science Foundation by grant number 2100-056986.99.

I am grateful to my parents for their patience and love.

Neuchâtel, Switzerland — Laura Elena Răileanu
May, 2002

Chapter 1

Introduction

In the last years, we have seen an explosive growth in our capabilities to both generate and collect data. Examples of this growth can be found in all sectors such as scientific, business and governmental organizations. Even though there is a large amount of available data, there is an obvious lack of valuable information. The development of new tools and techniques to extract the required information from huge amounts of data is one of the main challenges of the emerging field of Knowledge Discovery in Databases (KDD).

In [18], the *Knowledge Discovery in Databases* is defined as the non trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.

The KDD process includes several steps and, an important one is the *Data Mining* step, which consists of particular data mining algorithms that, under some acceptable computational efficiency limitations, produces a particular enumeration of patterns. The Data Mining component of the KDD process is mainly concerned with means by which patterns are extracted and enumerated from the data. Data mining is a fruitful area of research with an abundant wealth of practical applications.

The tasks well suited for data mining are: *classification*, *estimation*, *prediction*, *affinity grouping*, *clustering* and *description*.

Some of these tasks are best approached in a top-down manner called *hypothesis testing*. In hypothesis testing, a database past behavior is used to verify or disprove preconceived notions, ideas, and hunches concerning relationships in the data. Other tasks are best approached in a bottom-up manner called *knowledge discovery*. In knowledge discovery, no prior assumptions are made; the data is allowed to speak for itself. Knowledge discovery

comes in two flavors: directed and undirected. Directed knowledge discovery attempts to explain or categorize some particular data field. Undirected knowledge discovery attempts to find patterns or similarities among groups of records without use of a particular target field or collection of predefined classes. All of these activities fall under our definition of data mining.

An important aspect of the mining problem lies in the need to extend known techniques and tools in a way that they are robust enough to handle the characteristics of real world databases.

The algorithms for *classification* involve inducing a classification function (or a classifier in terms of values of "condition" attributes) that partitions a given set of tuples into meaningful disjoint subclasses, as defined by user or the values of some "decision" attributes. The simplest way to represent a *classifier* is a black box which produces a decision for every admissible pattern of data that is presented to it [66]. Classification algorithms discover patterns that distinguish tuples belonging to one concept from those belonging to other concepts [14].

Some purposes of classification [21] are:

- *Data simplification* Classification can be a means of reducing large amounts of data to manageable summary form. As the volume and complexity of data increase, the human brain becomes less able to hold in balance all the different factors that are relevant to the assessment of the data. Technological change, in the form of more powerful computers, is the driving force behind this problem. Summarizing data can help to detect any important relationships and pattern within the data. More efficient organization and retrieval of the information available is possible in this way.

- Another purpose of classification is the *prediction.* If data have been summarized in a manner which allows relationships to emerge more clearly, the results are often sufficiently striking to aid the investigator in making predictions, or in discovering some hypotheses to account for the form of the data.

- There are various properties which one could hope that classification would possess. These desiderata are likely to depend on the field of study and the use that will be

made of classification. In the addition to the aims of simplification and prediction are *stability* and *objectivity*.

The first requirement of stability can be interpreted in several ways. One could demand that the classification be little affected:

a) by small errors in recording the variables describing each object;

b) by the addition of a few new objects to the data set;

c) by the recording of a new variables for each object in the data set;

The second requirement of stability involves that the objects under study are a representative sample from some larger collection of objects; the third requirement involves an analogous assumption about variables.

Objectivity involves the repeatability of results.

In the literature, one way to classify *classification methods* [66] is:

- statistical pattern recognition: Bayesian classifiers, linear discriminants (normal distribution, logistic regression), nearest neighbor methods (absolute distance, Euclidean distance, various normalized distances);
- neural networks: perceptrons (least mean square learning systems), multilayer neural networks (back-propagation);
- machine learning methods: decision trees;

Decision trees and memory based reasoning are techniques well-suited to classification. Link analysis is also useful for classification in some circumstances.

A *decision tree* is a classifier constructed using a divide-and-conquer strategy, by selecting an attribute as the root of the tree and making branches for all different values of this attribute. If all examples at a particular node belong to one class this node becomes a leaf. Otherwise, we may select an attribute that does not occur on the path to the root, and create further branches for all its values.

The selection of a feature (attribute) as a node to partition data available to it is done using a criterion. In general, feature selection is a process that chooses an optimal subset of features according to a certain criterion.

Several advantages of decision tree based classification have been pointed out in the literature. We enumerate some of them:

- Knowledge acquisition from pre-classified examples circumvents the bottleneck of acquiring knowledge from a domain expert.
- The methods are non-parametric as opposed to inferential methods, trees can model a wide range of data distributions.
- Hierarchical decomposition implies better use of available attributes and computational efficiency in classification.
- Trees classifiers can treat uni-modal as well as multi-modal data in the same fashion as opposed to some statistical methods.
- Trees can be used in deterministic as well as in incomplete problems. In deterministic domains, the dependent variable can be determined from the independent variables, whereas in incomplete problems, it cannot.
- Trees perform classification by tests whose semantics are clear to domain experts.
- Computationally, the decision trees procedure is a relatively quick one.
- In many applications that require easily explained solution formats, tree induction is the procedure of choice.
- The attractiveness of the tree-based methods is due to the fact that, in contrast to neural networks, decision trees represent *rules* that can be easily understood and interpreted. A *production rule* [56] is represented as $L \longrightarrow R$ in which the left-hand side L is a conjunction of attribute-based tests and the right-hand side R is a class.

1.1 Our contributions

This thesis considers *decision trees*, which are a way to represent rules underlying data.

The selection of the attribute used in each node of the tree to split the data (split criterion) is crucial in order to correctly classify objects. The criterion specifies the details of measuring feature subsets. The choice of a criterion can be influenced by the purposes

of feature selection. We need evaluation of features. For example, the evaluation of a feature can be measured in terms of whether selected features help improve a classifier's accuracy and whether selected features help simplify the learned results so that they are more understandable. Different split criteria were proposed in the literature (Information Gain, Gini Index, etc.). It is not obvious which of the split criteria will produce the "best" decision tree for a given data set. A large amount of empirical tests were conducted in order to answer this question, and no conclusive results were found. No one made a formal analysis of these split criteria and that is why this work treats in detail these criteria.

Therefore, this thesis gives a formal analysis of the well-known split criteria: Gini Index and Information Gain, in order to achieve more *fundamental insights* on how to select split criteria. First, the set of the commonly used split criteria are expressed in *one single formalism* in order to make them comparable. Secondly, the newly formulated split criteria are compared on a theoretical base. The results of the *mathematical comparison* of these functions conduct us to say that there are definitively some differences in the attribute selection done by these two criteria. By this formal analysis, we were able to study the behavior of the Gini Index and Information Gain, to give an exact mathematical description of the situations when they are choosing the same test to split on and when they are choosing different tests. This allows us, without constructing decision trees, to decide for a given database if the Gini Index criterion and the Information Gain criterion select the same split attribute. It is important to emphasize that the number of the cases in which the selection of attribute done by the two criteria is not similar is very small. This permits us to affirm that, in general, the two criteria select identically.

The two well-known split criteria behave more or less equivalently. In the actual context of work, when we are confronted with *very large databases*, a study of these split criteria in relation with increasing sizes of data sets is useful. For this objective, a new family of split criteria (including the Gini Index and the Information Gain functions) is introduced. The new defined family of split functions is tested on data sets of different sizes. The tests show the sensitivity of the two popular split functions to the variation of the training set sizes and the predictable behavior of the simplest members of the introduced family of split functions.

The sensitivity of the behavior of the Gini Index and Information Gain functions in the relation with the size of the training sets, leads us to realize, in a third step, *a formal analysis of the relation between the size of a training set and the decision tree inferred from this training set. A new sampling procedure* that allows us to reduce the training set size, without encumbering the quality of the inferred decision trees is introduced. An exhaustive set of tests is conducted, to show that the theoretically proven properties are also valid in practice. The new proposed sampling procedure is applied to real data sets, and it is shown an almost perfect behavior of it in the sense, that the decision tree inferred from the whole database as well as the sampled database are almost always identical.

1.2 Overview of the thesis

The main topic of this thesis is the behavior of the split criteria used in decision tree induction in the context of large data sets.

Chapter 2 presents a survey of decision trees induction and discusses fundamental issues associated to it in the context of large data sets.

Chapter 3, begins with a presentation of the studies conducted by other researchers to analyze decision trees. We approach this issue in another manner: our contribution is the formal analysis of the existing decision tree criteria: Gini Index and Information Gain. We characterize mathematically how the two split criteria choose the split attributes. There are some differences in their choice, and this conducts us to analyze them in relation with the increasing size of databases.

In a context of large data sets, chapter 4, addresses another way to build better trees: by introducing a new family of split functions (including the well known split functions Gini Index and Information Gain) which are tested on data sets of different size.

Chapter 5, is dedicated to our new technique of sampling applied to large data sets which guarantees that the decision trees inferred from the whole database as well as from the sampled database are almost always identical. Theoretical analyzes and practical evaluations on real large data sets prove the validity of our new technique of sampling.

Finally, chapter 6 provides general conclusions and outlines the contribution of this thesis.

Chapter 2

Existing work on decision trees

This chapter begins with a rough introduction to inductive learning, followed by some definitions used in this work and a quick overview of the process of decision tree induction. Some of the issues related to the construction of decision trees and to our work are reminded. The methods used to evaluate the decision trees are presented. We list the other exploration methods before enumerating the applications of decision trees.

2.1 Introduction to the Inductive Learning

Learning is defined by [62] as "changes in a system that enables it to do the same task or tasks drawn for the same population more efficiently and more effectively the next time." A system can change in two ways:

(1) by acquiring new knowledge from external sources, or

(2) by modifying itself using its current knowledge more effectively.

Corresponding to these two types of changes, the learning can be divided into:

(1) *empirical learning* (or *inductive learning* or *similarity-based learning*) (accomplished by reasoning from external supplied examples to produce general rules or procedures; many algorithms operate by comparing the training examples to one another to find similarities), and

(2) *speedup learning* or *skill acquisition* (in which the system improves its performance by exploiting its current knowledge more effectively).

The empirical learning is subdivided into two types: (1.a) *supervised learning* and (1.b) *unsupervised learning.*

In the *supervised discovery* (1.a), the learning program is given examples of the form (x_i, y_i), and it is supposed to learn a function f, such as $f(x_i) = y_i$ for all i. Using the training examples the function is constructed to be used later to predict y values for new, previously unseen, x values. Each x value corresponds to a description of some object, situation or event and each y is a simpler description. Each possible value of y corresponds to a class, and the function f can be viewed as assigning each x to a class. For example, x could be a description of a client's recent purchases, and y the decision to send that customer a catalog or not; or x could be a record of a cellular-telephone call, and y the decision whether it is fraudulent or not. The function f can be viewed as a model, that can be used to classify unlabelled, previously unseen examples. The generated model can be evaluated in terms of accuracy (error rate), comprehensibility, compactness, learning speed, classification speed, etc.

The *unsupervised discovery* (1.b) is split off into clustering and discovery. The learning program is given a set of x_i values and asked to search for regularities that take the form of "clusters" of x_i values, and hence this task is called clustering. The discovery programs look for more complex relationship among the x_i values.

Several areas of human activity can involve *supervised machine learning*: predicting the use of a land, based on satellite images; assigning credit to individual basis of financial information; sorting letters based on machine-readable post-codes; preliminary diagnosis of a patient's disease; etc.

Throughout this thesis we use the following definitions:

Definition 2.1.1. An *information system* I, is a system $< U, A >$, where:
(1) $U = \{u_1, u_2, \ldots, u_{|U|}\}$ is a finite non-empty set, called the *universe* or *object space*; elements of U are called *objects*, or *examples*, or *instances*;
(2) $A = \{A_1, A_2, \ldots, A_{|A|}\}$ is also a finite non-empty set; elements of A are called *attributes*, or *variables*, or *features*;
(3) for every $A_j \in A$ there is a mapping A_j from U into some space $A_j : U \to A_j(U)$ and $A_j(U) = \{A_j(u) | u \in U\}$ is called the *domain* of the attribute A_j.

Definition 2.1.2. According to [21] and [68], the attributes are categorized into:
(1) *numeric*: the values taken by a numeric, or quantitative, attribute are real numbers. Sometimes a distinction is made between *ratio numeric* attributes and *interval numeric* attributes. For ratio attributes, the value "0" is unambiguously defined: if x_1 and x_2 are

two values taken by a ratio attribute, $\frac{x_1}{x_2}$ is well defined. When measuring distance between an object and itself forms a natural zero. For interval attributes the value zero is not well defined, but the difference $(x_1 - x_2)$ has meaning. The temperature of a patient is an example of an interval attribute, since it would not be useful to attempt to define "zero temperature" in this context.

(2) *nominal and ordinal*: each of these types of attribute comprises a set of discrete states such that each object belongs to one and only one state. Example is the "colour of eyes", this attribute having the states: "blue", "green", "brown", "black", etc. *Nominal* attributes are those for which no ordering can be placed on the different states. If two individuals are observed to belong to the states S_1, S_2, respectively, of a nominal attribute, then either $S_1 = S_2$ or $S_1 \neq S_2$; statements as "$S_1 < S_2$" have no meaning for nominal attribute. *Ordinal* attributes are those for which an underlying order can be imposed on the attributes states: the situation "$S_1 \neq S_2$" can be resolved into either "$S_1 < S_2$" or "$S_1 > S_2$". The attribute "temperature" having the states: "hot", "mild" and "cool" is an ordinal attribute.

Nominal attributes are sometimes called *categorical*, *enumerate*, or *discrete*. *Enumerated* is the standard term used in computer science to denote a categorical data type; however, the strict definition of the term-namely, to put into one-to-one correspondence with the natural numbers-implies an ordering, which is specifically not implied in the machine learning context. *Discrete* has also connotations of ordering because you often discretize a continuous, numeric quantity. Ordinal attributes are generally called *numeric*, or perhaps *continuous*, but without the implication of mathematical continuity. A special case of the nominal scale is the *dichotomy*, which has only two members-often designated as *true* and *false*, or *yes* and *no*. Such attributes are called sometimes *boolean*.

Definition 2.1.3. A *decision table* is an information system $< U, A >$ in which A is partitioned into two sets C and D: $A = C \cup D$ and $C \cap D = \emptyset$. The attributes in C are called *condition attributes* and attributes in D are called *decision attributes* or *target attributes*. The target attribute is assumed to take values in a predefined set of nominal or ordinal values. It is also referred to *class* or *label*.

Definition 2.1.4. A *data set* is a set of examples. We distinguish between the *training set* that is a set of labelled examples, and the *test set* that is a set of unlabelled examples.

Definition 2.1.5. A *model* is a mapping between the condition variables and the target variable. It is used to predict the target variable for unlabelled examples. Usually the target variable is referred as the *dependent* or the *response* variable, and all the other variables are referred to as *independent*, *explanatory*, or *predictor* variables.

Definition 2.1.6. A *learner* is any procedure used to build a model. In the context of this thesis we are referring the learners to computer programs that built the *model* from a *training set* without interaction with the users or other for of knowledge about the problem.

2.2 Basics of Decision Trees

Decision tree learning is one of the most widely used methods for inductive learning, the learning function corresponds to a decision tree. The decision trees are used to classify examples, more exactly to assign a new example on the basis of a set of measurements to one of a number of possible classes. The basis in the induction of a decision tree is the

training set, which consists of objects. Each object is described by a set of attributes and a class label. Each attribute measures some important feature of an object. The values of the attributes can be ordered or unordered. Each object of the training set belongs to one of a set of mutually exclusive classes. The induction task is to obtain classifications rules that can determine the class of any object based on its values of the attributes.

Definition 2.2.1. *A decision tree* [56] is a form of classifier, a structure that is either:
a *leaf*, indicating a class, or
a *decision node*, that specifies some test to be carried out on a single attribute value, with one branch and subtree for each possible outcome of the test.

The objects are classified by decision trees which sort them down from the *root* to some leaf node, which provides the classification (the class) of each object. An decision tree contains zero or more *internal nodes* and one or more *leaf nodes*. The internal nodes have two or more *child nodes*. Each non terminal node contains *a split* which specifies *a test* of some attribute, and each branch descending from that node corresponds to one of the possible values for this attribute. Each leaf node has its class label. A leaf node is said to be *pure* if all of its training examples are belonging to the same class.

Thus, an example is classified by starting with the root node, testing the attribute corresponding to the root node, then moving down the tree branch corresponding to the value of the attribute in the given example. This process continues until the example reaches a leaf node.

2.2.1 The Basic Decision Tree Learning Algorithm

The construction of a tree is called *tree induction*. Most algorithms that have been developed for learning decision trees are variations on a core algorithm that employs a top-down, greedy search through the space of possible decision trees. The ID3 algorithm (Quinlan 1986) and its successor C4.5 (Quinlan 1993) or the CART (Breiman 1984) are such examples. The original idea goes back to the skeleton of Hunt's method [56] for constructing a decision tree from a set $\mathcal{L}$ of training examples, examples belonging to the classes: $\{c_1, c_2, \ldots, c_k\}$.

- If $\mathcal{L}$ contains no cases, the decision tree will be a leaf; the associated class of this leaf have to be determined from information outside of $\mathcal{L}$.

- If $\mathcal{L}$ contains one or more examples, all belonging to the same class c_i, the decision tree will be a leaf identifying class c_i.

- If $\mathcal{L}$ contains examples of different classes, $\mathcal{L}$ is refined into subsets of examples belonging to the same class or heading towards a single class. A test based on a goodness measure and on a single attribute is chosen to split the examples of $\mathcal{L}$ in several subsets. The test has one or more mutually exclusive outcomes $O_1, O_2, \ldots, O_n$ corresponding to the values of the chosen attribute. $\mathcal{L}$ is split in $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$, where each $\mathcal{L}_i$ contains all the examples of $\mathcal{L}$ having the outcome O_i. The decision tree is formed by a decision node identifying the test, and one branch for each possible outcome. This procedure is recursively applied to each subset of training examples, so that the ith branch leads to the decision tree constructed from the subset $\mathcal{L}_i$.

2.3 Constructing the Decision Tree Classifier

The decision trees are constructed top-down, beginning with the question "which attribute should be tested at the root of the tree?" To answer this question, each instance attribute is evaluated using a criterion to determine how well it alone classifies the training examples. The attribute selected by the criterion is used as test at the root node of the tree. For each possible value of this attribute a branch is then created and, the training examples are sorted to the appropriate descendant node (i.e, down the branch corresponding to the example's value for this attribute). This process is repeated using the training examples associated with each descendant node to select the best attribute to test at that point of the tree.

The central choice in the algorithm of the construction of a decision tree is *the selection of the attribute to test at each node in the tree.* We would like to choose the attribute that is most useful to classify examples. What is a good quantitative measure of the worth of an attribute? Most of the methods described in literature agree that a split which maintains the classes' proportions in every subset has no utility, and a split in which each subset contains only examples of one class has maximum utility. Attribute evaluation criteria are heuristics reliable in the decision tree construction. Ben-Bassat [5] proposed an attribute evaluation criteria taxonomy. He divided the criteria into three, not necessarily distinct, categories:

attribute evaluation criteria derived from the information theory, criteria derived from distance measures and criteria derived from dependence measures. A detailed description of each category of attribute evaluation can be found in [39].

- *Criteria derived from information theory*: Examples of this variety are criteria based on Shannon's entropy. Tree construction by maximizing global *mutual information*, i.e, by expanding tree nodes that contribute to the largest gain in average mutual information of the whole tree, is explored in pattern recognition. Tree construction by locally optimizing *Information Gain*, the reduction in entropy due to splitting each individual node, is explored in pattern recognition, in sequential fault diagnosis and in machine learning [54]. In [34] the *G-statistic* is proposed for tree construction as well as for deciding when to stop.

- *Criteria derived from distance measures*: here "distance" refers to the distance between class probability distributions. The feature evaluation criteria in this class measure separability, divergence or discrimination between classes. A popular distance measure is the *Gini Index* of diversity. The Gini Index has been used for tree construction in statistics, pattern recognition and sequential fault diagnosis. The *twoing rule, Bhattacharya distance, Kolmogorov-Smirnoff distance* and the χ^2 *statistic* are some other distance-based measures that have been used for tree induction.

- *Criteria derived from dependence measures*: they use the statistical dependence between two random variables.

There exist several attribute selection criteria that do not clearly belong to any category in Ben-Basset's taxonomy. Quinlan and Rivest [57] suggested the use of *Risannen's minimum description length* for deciding which splits to prefer over others and also for pruning. Heath [22] used for oblique decision tree induction the *max minority* and the *sum minority* measures, representing the maximum and the sum of the number of misclassified points on either side of a binary split.

A consistent part of the work of this thesis is to bring into a common formalism the most popular decision tree criteria, in order to compare them theoretically. In the literature of speciality we can find only empirical comparisons between the decision trees inferred using

different criteria of split, no one did a "real" analysis. More details about the existing empirical studies can be found in the next chapter. Our formal analysis will permit us to formulate the conditions when the selection performed by the different split criteria is the same (independent of the use of an criterion or other). This theoretical evaluation leads us to introduce a *new family of split criteria.*

2.4 Right Sized Trees

The recursive partitioning method of inferring decision trees described in subsection 2.2.1 will continue to subdivide the set of training cases until each subset in the partitions contains cases of a single class, or until no test offers any improvement. While this is sometimes a reasonable strategy, in reality it can lead to difficulties when there is noise in the data, or when the number of training examples is too small. In either of these cases, trees that *overfit* the training examples are produced. A hypothesis overfits the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances (i.e., including instances beyond the training set). There are several approaches to avoid overfitting in decision tree learning:

- approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data, referred as *pre-pruning* methods, and
- approaches that allow the tree to overfit the data, and then post-prune the tree, referred as *post-pruning* methods.

All the methods make a trade-off between the size of the tree and an estimate of error rate. To determine the correct final tree size there are several approaches which:

- use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree,
- use all the data for training, but apply a statistical test to estimate whether expanding a particular node is likely to produce an improvement beyond the training set,
- use an explicit measure of the complexity for encoding the training examples and the decision tree, stopping growth of the tree when this encoding size is minimized.

Post-pruning is the most common method of pruning decision trees. A full tree is generated, overfitting the data, and then pruned back. Breiman et al. [7] suggested the following procedure: build the complete tree (a tree in which splitting no leaf node further will improve the accuracy on the training data) and then remove the subtrees that are not contributing significantly towards generalization accuracy. Their pruning method, *cost complexity* pruning has two stages: in the first one, a sequence of increasingly smaller trees are built on the training data. In the second stage, one of these trees is chosen as the pruned tree based on its accuracy on a *pruning set.* Another technique which predicts error rate of the tree and its subtrees using a new set of cases which is distinct from the training set is *reduced-error* pruning [55]. This family of techniques has the drawback that when data is scarce they are conducting to worse trees. To solve this, Breiman [7] proposed the *cross-validation* approach. In C4.5 Quinlan proposed another technique, *rule post pruning*, that uses only the training set from which the tree was built; to predict error rates he used a method called *pessimistic pruning* based on the confidence limits for the binomial distribution. A detailed description of the advantages and inconveniences of the approaches used to avoid overfitting in decision tree learning is given in [36].

2.5 Other Issues in Decision Trees

2.5.1 Feature Subset Selection and Subsample Selection

Tree constructions involves many issues other than finding good splits and knowing when to stop recursive splitting. Using algorithms for large databases is a classical scaling up problem. It is associated with the following characteristics of large databases:

- the size of database, e.g., millions of examples,
- the number of attributes describing the examples, e.g., hundreds or thousands,
- most often the features are continuous-valued, not discrete,
- features may be correlated.

The last problem, along with the second, can be addressed by performing feature selection using algorithms that are able to select the most significant features even when they are

highly correlated. The problem with the large number of attributes of a database can be solved performing feature selection as a preprocessing step. Another approach is to randomly choose disjoint subsets of features and run the algorithms on these reduced databases. Those giving best results should be kept. The best approach to solve the problem of having large numbers of attributes is to rank them and use only the features with highest rankings. A good description of the algorithms which can be used for this type of feature selection can be found in [11]. In [6], Blum and Langley review the problem of selecting relevant features. The feature selection methods are grouped into three classes: those that embed the selection within the basic induction algorithm, those that use feature selection to filter features passed to induction, and those that treat feature selection as a wrapper around the induction process.

Just as some attributes are more useful than others, some examples may be better for the learning process than others. The problem associated with the large size of a database can be addressed by splitting the database into several disjoint subsets (*simple random sampling*), and then by generating decision trees for each subset. It was shown that the accuracy of each generated decision tree is less than that of one that could have been generated from the entire data set. *Stratification* is a technique from a subfield of statistics called sample surveys. Subclasses of interest are identified beforehand, treated differently and reweighting is performed on samples. No major improvements were observed. Another approach to the same problem which is also related to the continuous-valued features, is to discretize them. Quinlan suggested *windowing*, a random training sample method, for his programs ID3 and C4.5 [54, 56]. A initially randomly chosen window can be iteratively expanded to include only the "important" training samples. Catlett [9] describes another method called *peepholing* designed for large training sets. The method works by examining random subsets of the larger sets of examples to be assessed. From these it is possible to eliminate attributes and a large fraction of their range of values.

Solid knowledge about the effect of sampling in the context of decision tree induction is very important, because the amount of available data is drastically increasing, and the existing induction tools are not able to handle large amounts of data. However the existing methods do not provide a one-shot sampling approach. To overcome the problem of large

data sets, in chapter 5, we propose *a new sampling method* which is efficient as shown by our theoretical analysis and also by the reported experiments on real databases. Differently to the other existing methods, our sampling technique is applied only once before beginning the decision tree induction (not for each node of the decision tree).

A more complete description of the problems that occur when dealing with large databases and the algorithms which can solve them can be found in [11].

2.5.2 Missing Values

In real world data sets, it is often the case that some attribute values are missing from the data. Several solutions were proposed in the literature:

- The simplest method is to replace the unknown value with the most common value for the attribute found in the training set.
- Another method is to consider the unknown value as another possible value of the attribute and when building the tree, each decision node can contain a branch for the case where the tested attribute takes an unknown value.
- In C4.5 a probability is assigned to each of the possible values. The probabilities are estimated on the basis of the observed frequencies of the possible values for the attribute in the examples at the current node. These fractional values are used to compute the Information Gain and can be further subdivided at subsequent branches of the tree if another missing attribute value is to be tested.
- CART uses another strategy: *surrogate splits*. Instead of keeping, at each node, only the attribute minimizing the impurity function, CART keeps a ranking of attributes producing a similar split. When classifying an example, if the value of the best attribute is unknown, CART will look for the attribute with a known value following the order given by the ranking.

2.5.3 Multivariate Splits

Usually the decision trees are univariate, i.e., theirs splits are constructed based on one attribute. Multivariate decision trees can use splits that contain more than one attribute

at each internal node. Most of the work on multivariate splits considered linear (oblique) trees: the tests are based on a linear combination of the attributes at some internal nodes. The first proposed oblique decision tree algorithm was CART-LC [7]. Another ones are: the Linear Machine Decision Trees (LMDT) proposed by Brodley and Utgoff [63, 8, 16], Simulated Annealing of Decision Trees (SADT) proposed by Heath, Kasif and Salzberg [22], and another alternative is offered by linear programming (LP). Another systems which performs oblique decision tree induction is the Oblique Classifier 1 (OC1) [40]. More details of these methods can be found in [39, 1].

2.5.4 Analyses

Several researchers have tried to evaluate the tree induction method itself, to precisely answer questions such as "it is possible to build optimal trees?", "how good are particular feature evaluation rules or pruning methods?". Our investigations are related to the second question, more precisely we made theoretical investigations in the choice of the attribute on which we split on, if the split criterion is the Gini Index or the Information Gain.

NP-hardness

Several aspect of optimal tree construction are known to be intractable. Hyafil and Rivest [24] proved that the problem of building optimal decision trees from decision tables, optimal in the sense of minimizing the expected number of tests required to qualify an unknown sample, is NP-hard. Usually the algorithms perform a one-step lookahead search. Once a decision is taken, it is never reconsidered. This hill-climbing search without backtracking has the risk of converging to locally optimal solutions that are not globally optimal. This strategy allows to build decision trees in time linear to the number of examples.

The articles [39, 36] discuss also some other issues involved in construction of decision trees like: improving on greedy induction, use of fuzziness to remove data fragmentation and class overlap, incorporating attribute measurement costs and misclassification costs into tree construction, estimating class probabilities from trees, use of multiple trees to reduce variance and incremental induction of trees.

2.6 Methods for Evaluation Decision Trees

An important point is the design of the empirical evaluation of decision trees. In this section we review the common practice and present the methodology of empirical evaluation followed in this dissertation. This methodology will be applied in chapter 4 where we will evaluate the quality of the inferred decision trees using our new family of split functions (compared with the trees constructed the classical split functions: Gini Index and Information Gain), and respectively in chapter 5 where we will test the validity of our new technique of sampling applied to large data sets.

In the machine learning literature the parameters considered frequently include error rate, comprehensibility, compactness, learning speed, classification speed, etc.

Definition 2.6.1. The *error rate* is computed as the ratio between the number of misclassifications and the total number of examples. The *accuracy* is the proportion of objects that are correctly classified by a decision tree.

Definition 2.6.2. We need to distinguish between two different error rates. One, denoted as *resubstitution error* estimates the error rate from the training set, and the other, *the generalization error rate*, estimates the error from an independent test set.

The evaluation of the rules generated by decision trees is done using the the concepts of *completeness* and *consistency*. These concepts have been explored by researchers in many areas, but no single terminology to describe them has been established. A detailed overview of them is given in [26].

Let P and N respectively denote the number of positive and negative examples in the training set. Let p and n respectively represent the number of positive and negative examples covered by a given rule R.

Definition 2.6.3. The *coverage of* R is the number of cases that satisfy the rule: $\text{cov}(R) = p + n$, where p may be referred to as *support* or *positive coverage* and n may be referred to as *negative coverage*.

Definition 2.6.4. For the given rule, the ratio $\frac{p}{P}$, denoted $\text{compl}(R)$, is called the *completeness* or *relative coverage* of R.

Definition 2.6.5. The ratio $\frac{p}{p+n}$ denoted $\text{cons}(R)$, is called the *consistency* or *training accuracy* of R, and $\frac{n}{p+n}$ denoted $\text{inc}(R)$, the *inconsistency* or *training error rate*.

If $\text{compl}(R) = 100\%$ then R is *complete cover* of the training examples. If $\text{inc}(R) = 0\%$, then R it is a *consistent cover*.

We used in this thesis the Quinlan's error rates definitions for decision trees [56]. More precisely: when N training cases are covered by a leaf, E of them incorrectly, the resubstitution error rate for this leaf is $\frac{E}{N}$. The sum of the resubstitution error rates at the leaves, divided by the number of cases in the training set represents the *error rate on the training data.* If we use the test data in the same manner, we obtain the *error rate on the test data.* If pruning is performed, error estimates for leaves and subtrees are computed assuming that they were used to classify a set of unseen cases of the same size as the training set. So a leaf covering N training cases with a predicted error rate of $U_{CF}(E, N)$ (the upper limit of a binomial distribution corresponding to a given confidence level CF) would give rise to a predicted $N \times U_{CF}(E, N)$ errors. Similarly, the number of predicted errors associated with a (sub)tree is the sum of the predicted errors of its branches. The sum of the predicted errors at the leaves, divided by the number of cases in the training set, provides the *estimate of the error rate of the pruned tree on unseen cases.* The Quinlan's *confusion matrix* is useful for characterizing accuracy.

Definition 2.6.6. The size of the model (number of nodes, number of leaves or depth of a tree) represents its *degree in compactness.*

Definition 2.6.7. The *learning time* is the time needed by the algorithm to build the classifier. The *classification time* or *testing time* represents the time needed to classify a new example.

As all the algorithms used in this dissertation are relatively fast, we exclude the learning time or the classification time from our experimental comparisons.

Estimating the Error Rate

To evaluate a given classifier, which is in our case a decision tree, one should construct training and test sets. This can be done by drawing randomly a very large a very large independent set of examples from the same population. The decision tree generated from the training set is used to classify each of the examples in the test set. If the number of examples is large enough, the proportion of misclassified examples gives an unbiased estimate of the error rate. In reality, it is difficult sometimes to obtain large training and test sets, so the available data, are used both to build the decision tree and to estimate the error rate. The *resubstitution* estimate is used, but the problem is that it is a biased

estimator. This is why in practice the methods of evaluation split the available data into two samples: the training set and the test set. Several methods exist for obtaining samples.

- *Bootstrap*: it is a procedure of sampling with replacement from the available data. If n examples are available, a sample of size n is drawn with replacement to form the training set. The test set contains the set of examples that do not appear in the training set. The learning algorithm generates a model from the training set. This model is used to classify the examples of the test set. This process is repeated several times (ten times), each time using a different bootstrap sample. The estimate of the error rate on the given data set is the average of the error rates in each bootstrap sample.

- *n-fold Cross-Validation*: the examples are randomly placed into one of the n partitions. A sample is formed by setting aside one of n partitions as the test set, and the remaining partitions make up the training set. The model is used to classify the examples of the test set. This whole process is repeated for each of the n training-test set. The estimate of the error rate on the given data set is the average of the error rate in each fold. A variant of this method referred as *stratified cross validation*, obtains partitions of the data that maintain the distribution of the classes in the original data.

- *Monte-Carlo Cross-Validation*: is a special case of n-fold Cross-Validation. A percentage of the available examples (usually 2/3) is randomly placed in the training set and the remaining examples are placed in the test set. After learning takes place on the training set, the model is estimated on test set. This whole process is repeated for many training-test pairs. The estimate of the error rate on the given data set is the average of the error rates in each run.

The method we have chosen in our evaluations is a 10-fold stratified cross validation, as in the literature it is emphasized as being the best one.

2.7 Other Exploration Methods

There exist several alternatives to decision trees to explore data, such as: neural networks, nearest neighbor methods, regression analysis, genetic classifiers, decision tables, decision

rules, linear discriminant analysis. Researchers have compared these methods to decision trees on specific problems. A survey of these comparisons and the results obtained by the researchers can found in [39].

2.8 Applications of Decision Trees

Decision trees are one of the most commonly used algorithms, both in real world applications and in academic research. This is due to the positive points referred to in the literature; some of their characteristics have been already point up in the introduction. The application areas are numerous: in agriculture, astronomy, biomedical engineering, control systems, financial analysis, manufacturing and production, medicine, molecular biology, object recognition, pharmacology, physics, plant diseases, power systems, remote sensing, software development, text processing. The works dedicated to applications of decision trees in real world are listed in [39]. In [17, 18] you can also find several applications.

Chapter 3

Uniform Formalization of Split Criteria and Theoretical Comparison between them

In this chapter, we first focus our attention on the existing decision split criteria and on the research directions consecrated to this essential step of decision tree algorithm. The fundamental concepts used to introduce the decision split functions are presented in an uniform way. Secondly, we use a common formalism to present the best known decision split criteria: Index of Gini, Information Gain, Twoing Rule, Max Minority, Sum Minority, Sum of Variances, Minimum Message Length. This formalism permits to introduce a formal methodology, which allows us to analytically compare multiple split criteria, and thus to present fundamental insights into the decision process. As an illustration, in the third part of this chapter, we give a theoretical characterization of the Gini Index split criterion (the conditions when an attribute is selected as test by the Gini Index function and when not), respectively of the Information Gain criterion followed by the comparison between them. We establish the cases (the conditions to be satisfied by the database's parameters) in which they are choosing the same attribute to split on and the cases in which they are choosing different attributes to split on.

3.1 Finding Splits to Construct Decision Tree: State of the Art

The current work in the field of the construction of decision trees that is related to our work can be divided into three groups.

The early work in the field of decision tree construction focused mainly on the definition and on the realization of classification systems. Such systems have intensively been developed since the 1980's [38, 58]. In the 1980's, Breiman et al.'s book [7] on classification and regression trees (CART) and Quinlan's work on ID3 [53, 54] provided the foundations for what has become a large body of research on one of the central technique of experimental machine learning. Many variants of decision tree algorithms have been introduced in the last decade. The majority of the work has been concentrated on decision trees in which each node checks the value of a single attribute [7, 54, 56]. Quinlan initially proposed decision trees for classification in domains with symbolic-valued attributes [54], and latter extended them to numeric domains [55]. When the attributes are numeric, the tests have the form $A_i > k$, where A_i is one of the attributes of an example and k is a constant. Researchers have also studied decision trees in which the test at a node uses Boolean combinations of attributes [42, 43, 59] or linear combinations of attributes [39]. All these systems are using different measures of impurity / entropy / goodness [32] to select the split attribute in order to construct the decision tree. We enumerate seven examples of such measures: the Information Gain [56], the Gini Index [7], the Twoing Rule [7], Max Minority [22], Sum Minority [22], Sum of Variances [40], MML [65].

Given the large number of evaluation measures, a natural concern is to decide their relative effectiveness in constructing "good" trees. That is the reason for which, the second part of the work in this field is dedicated to the analysis and to the comparison of different decision tree construction algorithms. Comparisons of different methods are interesting to determine which method is suitable for a given situation. Evaluations in the direction of deciding the effectiveness of the decision split criteria have been predominantly empirical. In spite of a large number of comparative studies, very few so far have concluded that a particular feature evaluation rule is significantly better than others. Baker and Jain [4] reported experiments comparing eleven feature evaluation criteria and concluded that the

feature rankings induced by various rules are very similar. Several feature evaluation criteria are compared using simulated data by Moshe Ben-Bassat in [5], on a sequential, multi-class classification problem. The conclusions are that no feature selection rule is consistently superior to the others, and that no specific strategy for alternating different rules seems to be significantly more effective. Mingers [35] compared several attribute selection criteria, and concluded that the tree quality does not seem to depend on the specific criterion used. Babic [3] compared ID3 and CART for two clinical diagnosis problems. Miyakawa [37] compared three activity-based measures, Q, O, and $loss$ both analytically and empirically. He showed that Q and O do not choose non-essential variables at tree nodes, and that they produce trees that are 1/4th the size of the trees produced by $loss$. Fayyad and Irani [32] showed that their measure C-SEP, performs better than Gini Index and Information Gain for specific types of problems. Several researchers pointed out that Information Gain is biased towards attributes with a large number of possible values. Mingers [34] compared Information Gain and χ^2 statistic for growing the tree as well as for stop splitting. He concluded that χ^2 corrected Information Gain's bias towards multivalued attributes. Quinlan [56] suggested Gain Ratio as a remedy for the bias of Information Gain. Mantaras [13] argued that Gain Ratio had its own set of problems, and suggested information theory based distance between partitions for tree constructions. White and Liu [67] present experiments to conclude that Information Gain, Gain Ratio and Mantara's measure are worse than a χ^2 based statistical measure, in terms of their bias towards multiple-valued attributes. J. Gama [19] (Esprit Project 5170 StatLog (1991-1994)) tried to predict the error rate of a particular classification algorithm and he indicated that no single method can be considered better than the others. About 20 different algorithms were evaluated on more than 20 different data sets. Kononenko [27] pointed out that Minimum Description Length based feature evaluation criteria have the least bias towards multi-valued attributes. In [28] twenty-two decision tree and two neural network algorithms are compared in terms of classification accuracy, training time, and number of leaves. In a recent paper, [64], the authors proposed a measure for the distance between the bias of two evaluation metrics and gave numerical approximations of it.

Several researchers have tried to evaluate the tree induction method itself, to answer

questions such as: is it possible to build optimal trees? How good are particular heuristics? Several aspects of optimal tree construction are known to be intractable. Hyafil and Rivest [24] proved that the problem of building optimal decision trees from decision tables, optimal in the sense of minimizing the expected number of tests required to classify an unknown sample is NP-Complete.

Finally there is a third family of work that relates to our interest. This work is connected to the Rough Set theory, which is an important approach characterizing classification from the viewpoint of set theory. It is a mathematical approach to imprecision, vagueness and uncertainty in data analysis. The basic operations of the Rough Set theory (the lower and the upper approximations) are used to discover fundamental patterns in data. This concept was proposed to solve basic problems, such as: description of set of objects in terms of attribute values, dependencies (full or partial) between attributes, reduction of attributes, significance of attributes. These are exactly the same problems put by classification in general, and by decision trees construction in particular. There exist decision systems based on the Rough Set formalism [29, 47, 69]. Most of this systems are build on top of the basic formalism introduced by Pawlak [44, 45, 46, 49, 29]. All this systems are using the simple roughness function given by Pawlak [48] as split criteria.

The existing decision tree work focused on the comparisons between the different systems seems to be almost empirical. It is a relatively hard task to compare the existing systems as they evolved from different backgrounds; some are founded in information theory, others in discriminant analysis, others are based on encoding techniques etc. Theoretical comparisons are very difficult as no common formalism was used. Therefore, most of the work went into *empirical* comparison of the different systems (see second group of related work). The results of these projects are very helpful for practical purposes, but do not really improve the *fundamental* understanding of the split criteria. It is not known how the different criteria relate to each other. Are they fundamentally all the same? Are there major differences? In order to answer these questions we will have to find a basic formalism that allows expressing the criteria. A clean formulation of the investigated split criteria is the key for a well-founded formal and empirical analysis.

However, a thorough understanding of the behavior of the split functions demands an

analytical and direct comparison between them, without using any other external measure. Our contribution is to introduce a formal methodology, which allows us to analytically compare multiple split criteria. This permits us to present fundamental insights into the decision process. Furthermore, we are able to present a formal description of how to select between split criteria for a given data set. As an illustration we apply the methodology to the two widely used split criteria: Gini Index and Information Gain. Thus, our work tries to overcome the lack of basic understanding of the split criteria selection process, by combining a proper formulation of the successfully used split criteria, followed by a formal and an experimental comparison.

In the following four sections we present notations and definitions necessary to introduce a formalism in which all the split criteria are uniformly presented. We give a detailed presentation of the Gini Index and Information Gain criteria using this formalism, and we briefly introduce some others.

3.2 Notations

To realize a theoretical analysis we begin by introducing some notations and definitions. Let $\mathcal{L}$ be a learning sample, $\mathcal{L} = \{(x_1, c_1), \ldots, (x_{\|\mathcal{L}\|}, c_J)\}$. We denote by $\|\mathcal{L}\|$ the number of objects in $\mathcal{L}$. Let x_i be a measurement vector and we have $\forall i \in \{1, \ldots, \|\mathcal{L}\|\}$, $x_i \in \mathcal{X}$, $\mathcal{X}$ being the measurement space. The set of classes is $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$, so $\forall i \in \{1, \ldots, J\}$, $c_i \in \mathcal{C}$. The prior probability that an object belongs to a given class c_i, is given by $p(c_i) = \frac{\|c_i\|}{\|\mathcal{L}\|}$. Given a test T, with n possible outcomes, we denote by t_i the set of the objects in $\mathcal{L}$ having the outcome i. The probability that the test T has the outcome i is estimated by $p(t_i) = \frac{\|t_i\|}{\|\mathcal{L}\|}$. We denote by $\|c_i, t_j\|$ the number of objects of $\mathcal{L}$ that lay in the class c_i and have the outcome j for the test T. The probability that an object lays in c_i and has the outcome j is given by $p(c_i, t_j) = \frac{\|c_i, t_j\|}{\|\mathcal{L}\|}$. The conditional probability, $p(c_i|t_j)$, that an object lays in the class c_i, under the condition that the test T has the outcome j, is estimated by the Bayes formula: $p(c_i|t_j) = \frac{p(c_i, t_j)}{p(t_j)}$. Obviously we have: $\sum_{i=1}^{k} p(c_i) = 1$, $\sum_{i=1}^{k} p(c_i|t_j) = 1$, $\forall j \in \{1, \ldots, n\}$ and $p(c_i)$, $p(c_i|t_j)$, $p(t_i) \in [0, 1]$, $\forall j \in \{1, \ldots, n\}$ and $\forall i \in \{1, \ldots, k\}$.

3.3 Definition of the Gini Index Function

In [7] the binary tree structured classifiers are constructed by repeated splits of subsets of $\mathcal{L}$ into two descendant subsets, beginning with $\mathcal{L}$ itself. To determine the binary splits of $\mathcal{L}$ into smaller and smaller sets, we have to select each split of a subset so that the examples in each of the descendent subsets are "purer" than the data in the parent subset. The idea is to introduce a measure of impurity (impurity function), the node impurity is largest when all examples classes are equally mixed together in it, and smallest when the node contains examples belonging to only one class. Thus, it was introduced the goodness of split criterion which is derived from the notion of impurity function.

Definition 3.3.1. An *impurity function* is a function ϕ defined on the set of all k-tuples of numbers $(p(c_1), p(c_2), \ldots, p(c_k))$ satisfying $p(c_i) \geq 0$, $\forall i \in \{1, \ldots, k\}$ and $\sum_{1=1}^{k} p(c_i) = 1$ with the following properties:

1) ϕ achieves its maximum at the point $(\frac{1}{k}, \frac{1}{k}, \ldots, \frac{1}{k})$;
2) ϕ achieves its minimum at the points $(1, 0, \ldots, 0)$, $(0, 1, \ldots, 0), \ldots,$ $(0, 0, \ldots, 1)$;
3) ϕ is a symmetric function of $(p(c_1), p(c_2), \ldots, p(c_k))$.

Definition 3.3.2. Given an impurity function ϕ, *the impurity measure of any node* t is defined by:

$$i(t) = \phi(p(c_1|t), p(c_2|t), \ldots, p(c_k|t)).$$

Definition 3.3.3. If a split s in a node t divides all examples into two subsets t_1 and t_2 of proportions p_1 and p_2, *the decrease of impurity* is defined as:

$$\Delta i(s, t) = i(t) - p_1 i(t_1) - p_2 i(t_2).$$

Definition 3.3.4. The *goodness of split* $\phi(s, t)$ is defined as $\Delta i(s, t)$.

Definition 3.3.5. If a test T is used in a node t and this test is based on an attribute having n possible values, the *the impurity measure of any node* t and *the decrease of impurity* are generalized as follows:

$$i(t) = \phi(p(c_1|t), p(c_2|t), \ldots, p(c_k|t))$$

$$\Delta i(s, t) = i(t) - \sum_{j=1}^{n} p(t_j) i(t_j).$$

Definition 3.3.6. Breiman adopts in his work the *Gini diversity Index* which has the following form:

$$\phi(p(c_1|t), p(c_2|t), \ldots, p(c_k|t)) = \sum_{i=1}^{k} \sum_{j=1, j \neq i}^{k} p(c_i|t) p(c_j|t) = 1 - \sum_{i=1}^{k} (p(c_i|t))^2.$$

In a node t an impurity function based on the Gini Index criterion assigns an example to a class c_i with the probability $p(c_i|t)$. The estimated probability that the item is actually in

class j is $p(c_j|t)$. Therefore, the estimated probability of misclassification under this rule is the Gini Index: $i(t) = \sum_{i=1}^{k} \sum_{j=1, j\neq i}^{k} p(c_i|t)p(c_j|t) = 1 - \sum_{j=1}^{k} (p(c_j|t))^2$.
This function can also be interpreted in terms of variance. In a node t we assign to all examples belonging to class c_j the value 1, and to all other examples the value 0. The sample variance of these values is: $p(c_j|t)(1 - p(c_j|t))$. There are k classes, thus the corresponding variances are summed together: $i(t) = \sum_{j=1}^{k} p(c_j|t)(1 - p(c_j|t)) = 1 - \sum_{j=1}^{k} (p(c_j|t))^2$.

Definition 3.3.7. Having a test T with n outcomes the goodness of the split is

$$\text{gini}(T) = 1 - \sum_{i=1}^{k} (p(c_i))^2 - \sum_{i=1}^{n} p(t_i) \sum_{j=1}^{k} p(c_j|t_i)(1 - p(c_j|t_i))$$

The Gini Index criterion selects a test that maximizes this function.

3.4 Definition of the Information Gain Function

The Information Gain function [56] has its origin in the information theory. It is based on the notion of entropy, which characterizes the impurity of an arbitrary set of examples. The information conveyed by a message depends on its probability and can be measured in bits as the negative logarithm to the base 2 of that probability. If we randomly select an example of a set the probability that it belongs to the class c_j is equal to $p(c_i) = \frac{\|c_i\|}{\|\mathcal{L}\|}$, and the amount of information the message "it belongs to c_j" conveys is $-\log_2(p(c_i))$.

Definition 3.4.1. The expected information provided by a message in respect to the class membership can be expressed as:

$$\text{info}(\mathcal{L}) = -\sum_{i=1}^{k} p(c_i) \log_2(p(c_i))$$

The quantity info($\mathcal{L}$) measures the average amount of information needed to identify the class of a case in $\mathcal{L}$. This quantity is also known as the *entropy of the set* $\mathcal{L}$ relative to the k-wise classification.

The logarithm is still base 2 because entropy is a measure of the expected encoding length measured in bits. We will consider a similar measurement after $\mathcal{L}$ has been partitioned into $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$ in accordance with the n outcomes of a test T. The expected information requirement is the weighted sum over the subsets: $\text{info}_T(\mathcal{L}) = \sum_{i=1}^{n} p(t_i)\text{info}(\mathcal{L}_i)$, info($\mathcal{L}_i$) being the amount of information needed to identify the class of a case in $\mathcal{L}_i$.

Definition 3.4.2. The information gained by partitioning $\mathcal{L}$ in accordance to the test T is measured by the quantity: $\text{gain}(T) = \text{info}(\mathcal{L}) - \text{info}_T(\mathcal{L})$. We can rewrite the

$$\text{gain}(T) = -\sum_{i=1}^{k} p(c_i)\log_2(p(c_i)) + \sum_{i=1}^{n} p(t_i)\sum_{j=1}^{k} p(c_j|t_i)\log_2(p(c_j|t_i)))$$

The Information Gain criterion selects a test that maximizes the Information Gain function.

3.5 Other Split Criteria

This section presents some of the best known and most used criteria in the decision tree construction using our formalism.

The Twoing Criterion

The Twoing Criterion was introduced by Breiman [7] to improve decision tree induction when there are a relatively large number of classes. We have a test based on an attribute having two possible values ($n = 2$). The Twoing Rule is defined as:

$$\begin{aligned} \text{twoing}(T) &= p(t_1)p(t_2)\left(\sum_{i=1}^{k}|p(c_i|t_1) - p(c_i|t_2)|\right)^2 = \\ &= p(t_1)(1-p(t_1))\left(\sum_{i=1}^{k}|p(c_i|t_1) - p(c_i|t_2)|\right)^2 \end{aligned} \tag{3.5.1}$$

If there are only two classes ($k = 2$) it is easy to show that the Twoing Rule corresponds to the Gini Index function.

The Max Minority Criterion

Max Minority is based on the number of misclassified objects on either side of a binary split. It was used by [22] for oblique decision tree induction.

$$\begin{aligned} \text{Minority}(T_L) &= \sum_{i=1, i\neq \max|p(c_i|t_1)p(t_1)|}^{k} |p(c_i|t_1)p(t_1)| \\ \text{Minority}(T_R) &= \sum_{i=1, i\neq \max|p(c_i|t_2)p(t_2)|}^{k} |p(c_i|t_2)p(t_2)| \\ \text{Max Minority}(T) &= \max\{\text{Minority}(T_L), \text{Minority}(T_R)\} \end{aligned} \tag{3.5.2}$$

The Sum Minority Criterion

Sum Minority is the sum of the $Minority(T_L)$ and $Minority(T_R)$ which are defined as for the *Max Minority* measure. The *Sum Minority* was also used by [22].

$$\text{Sum Minority}(T) = \text{Minority}(T_L) + \text{Minority}(T_R) \tag{3.5.3}$$

The Sum of Variances Criterion

The definition of the *Sum of Variances* measure given in [40] is:

$$\begin{aligned}
\text{Variance}(T_L) &= \sum_{i=1}^{k}\left(p(c_i,t_1)-\sum_{j=1}^{k}\frac{p(c_i,t_1)}{p(t_1)}\right)^2 \\
\text{Variance}(T_R) &= \sum_{i=1}^{k}\left(p(c_i,t_2)-\sum_{j=1}^{k}\frac{p(c_i,t_2)}{p(t_2)}\right)^2 \\
\text{Variance}(T) &= \text{Variance}(T_L)+\text{Variance}(T_R)
\end{aligned} \tag{3.5.4}$$

The MML Criterion

The *MML* was introduced by [65]. From Shannon's information theory we know that in an optimal code, the message length of an event E, MsgLen(E), where E has the probability $P(E)$, is given by $\text{MsgLen}(E) = -\log_2(E)$. For a hypothesis H and data D, the Bayes formula [50] gives: $P(H \cap D) = P(H) \cdot P(D|H) = P(D) \cdot P(H|D)$ where:

1. $P(H)$ represents the prior probability of hypothesis H (it represents $p(t_j)$),
2. $P(H|D)$ represents the posterior probability of hypothesis H (it represents $p(t_j|c_i)$),
3. $P(D|H)$ represents the likelihood of the hypothesis (it represents $p(c_i|t_j)$).

Applying the function $-\log_2$ to the Bayes formula, we obtain:

$$\text{MsgLen}(H \cap D) = \text{MsgLen}(H) + \text{MsgLen}(D|H) = \text{MsgLen}(D) + \text{MsgLen}(H|D).$$

In inductive inference one normally wants the hypothesis H with the largest posterior probability. MsgLen(H) can usually be estimated well and MsgLen($D|H$) can also be calculated. We may maximize the posterior probability of a model, $P(H|D)$, by maximizing $P(H) \cdot P(D|H)$, which is equivalent to minimize the message length:

$$\begin{aligned}
\text{MML}(T) &= -\log_2(P(H)\cdot P(D|H)) = -\log_2(P(H)) - \log_2(P(D|H)) \\
&= -\log_2(p(t_j)) - \log_2(p(c_i|t_j)) \\
&= \text{MsgLen}(H) + \text{MsgLen}(D|H)
\end{aligned} \tag{3.5.5}$$

After presenting all these criteria using a single formalism, we are able to introduce a formal methodology which is useful to compare analytically any pair criteria.

The next section is dedicated to the presentation of this methodology applied to Gini Index and Information Gain criteria.

3.6 Theoretical Analysis of the Gini Index and Information Gain Criteria

Fundamental knowledge about split criteria can be found by performing an analytical comparison between them. The selection of an attribute at each node of a tree is dictated by the split criterion. In practice it was noticed that different decision trees are obtained by using one or other split criterion. So, we have to compare locally the attribute selection performed by one or other criterion. We concentrate our attention to the Gini Index and Information Gain criteria. The same methodology which we will present here can be applied to analytically compare any other pair of split criteria.

The test selected to split a given node is that one which maximizes the Gini Index function, respectively the Information Gain function. So, the selected test by these criteria, T^*, will satisfy: $\text{gini}(T^*) = \max_{\forall\ T\ possible\ test} \text{gini}(T)$ and, respectively, $\text{gain}(T^*) = \max_{\forall\ T\ possible\ test} \text{gain}(T)$. So we have:

$$\text{gini}(T^*) \geq \text{gini}(T), \forall\ T\ possible\ test \tag{3.6.1}$$

and, respectively,

$$\text{gain}(T^*) \geq \text{gain}(T), \forall\ T\ possible\ test. \tag{3.6.2}$$

To obtain a characterization of the widely used Gini Index and Information Gain criteria, and to compare them, we restraint them, to the situation in which we have only two possible outcomes for the test T, $n = 2$, and two possible classes $k = 2$. Therefore, we have:

$$\text{gini}(T) = 1 - \sum_{i=1}^{2}(p(c_i))^2 - \sum_{i=1}^{2} p(t_i) \sum_{j=1}^{2} p(c_j|t_i)(1 - p(c_j|t_i)) \tag{3.6.3}$$

and, respectively

$$\text{gain}(T) = -\sum_{i=1}^{2} p(c_i)\log_2(p(c_i)) + \sum_{i=1}^{2} p(t_i) \sum_{j=1}^{2} p(c_j|t_i)\log_2(p(c_j|t_i)) \tag{3.6.4}$$

For simplicity we denote by: $x = p(c_1)$, $r = p(t_1)$, $p = p(c_1|t_1)$ and $q = p(c_1|t_2)$. We have: $1 - x = p(c_2)$, $1 - r = p(t_2)$, $1 - p = p(c_2|t_1)$ and $1 - q = p(c_2|t_2)$. Using these notations and some simple calculations we rewrite the Gini Index function and the Information Gain function as:

$$\text{gini}(T) = 2x(1 - x) - 2rp(1 - p) - 2(1 - r)q(1 - q) \tag{3.6.5}$$

and, respectively

$$\begin{aligned} \text{gain}(T) &= -x\log_2(x) - (1-x)\log_2(1-x) + r[p\log_2(p) + (1-p)\log_2(1-p)] \\ &\quad +(1-r)[q\log_2(q) + (1-q)\log_2(1-q)] \end{aligned} \tag{3.6.6}$$

where $x, p, q \in (0,1)$ and $r \in [0,1]$.

Let us suppose we have two tests, T, T' (based on two different attributes) which are used to split a given node. Now we analyze if the Gini Index criterion and the Information Gain criterion will select the same test. If this is not the case, we would like to know under which conditions the two criteria select differently.

First we will write the Gini Index (Information Gain) functions for the tests T, T':

$$\text{gini}(T) = 2x(1-x) - 2rp(1-p) - 2(1-r)q(1-q) \tag{3.6.7}$$

$$\text{gini}(T') = 2x'(1-x') - 2r'p'(1-p') - 2(1-r')q'(1-q') \tag{3.6.8}$$

and, respectively

$$\begin{aligned} \text{gain}(T) &= -x\log_2(x) - (1-x)\log_2(1-x) + r[p\log_2(p) + (1-p)\log_2(1-p)] \\ &\quad +(1-r)[q\log_2(q) + (1-q)\log_2(1-q)] \end{aligned} \tag{3.6.9}$$

$$\begin{aligned} \text{gain}(T') &= -x'\log_2(x') - (1-x')\log_2(1-x') + r'[p'\log_2(p') + (1-p')\log_2(1-p')] \\ &\quad +(1-r')[q'\log_2(q') + (1-q')\log_2(1-q')] \end{aligned} \tag{3.6.10}$$

where $x, p, q, p', q' \in (0,1)$ and $r, r' \in [0,1]$.

We observe that

$$x = x' \tag{3.6.11}$$

as $x = p(c_1) = \frac{\|c_1\|}{\|\mathcal{L}\|} = x'$. This probability remains constant, independently of the selected test. The number of examples belonging to the class c_1 and to the class c_2 respectively, remains constant, independently of the selected test, and therefore, the following relation holds:

$$r(p-q) + q = r'(p'-q') + q' \tag{3.6.12}$$

r relates to r', p, q, p', q' as it follows:

$$r = \frac{r'(p'-q') + q' - q}{p-q},\ p \neq q, \tag{3.6.13}$$

and r' relates to r, p, q, p', q' as it follows::

$$r' = \frac{r(p-q)+q-q'}{p'-q'},\ p' \neq q'. \tag{3.6.14}$$

The cases $p = q$, $p' = q'$, and $q = q'$ will be treated separately in the section 3.7.

Furthermore, the following conditions must be satisfied:

$$r' \geq 0 \Leftrightarrow \frac{r(p-q)+q-q'}{p'-q'} \geq 0,\ p' \neq q' \tag{3.6.15}$$

$$r \geq 0 \Leftrightarrow \frac{r'(p'-q')+q'-q}{p-q} \geq 0,\ p \neq q \tag{3.6.16}$$

$$r' \leq 1 \Leftrightarrow r'-1 \leq 0 \Leftrightarrow \frac{r(p-q)+q-p'}{p'-q'} \leq 0,\ p' \neq q' \tag{3.6.17}$$

$$r \leq 1 \Leftrightarrow r-1 \leq 0 \Leftrightarrow \frac{r'(p'-q')+q'-p}{p-q} \leq 0,\ p \neq q \tag{3.6.18}$$

$$p, q, p', q' \in [0,1] \tag{3.6.19}$$

The difference between the Gini Index functions corresponding to the tests T and T' can be written using (3.6.12) as:

$$\begin{aligned}
\text{gini}(T) - \text{gini}(T') &= 2r'p'(1-p') + 2(1-r')q'(1-q') - 2rp(1-p) - 2(1-r)q(1-q) \\
&= 2(r'q'^2 - r'p'^2 + r'p' - r'q' - q'^2 + q') - 2(rq^2 - rp^2 + rp - rq - q^2 + q) \\
&= 2[r'(q'-p')(q'+p') + r(p-q)(p+q) + (q-q')(q+q')] \\
&= 2[r(p-q)(p+q-p'-q') + (q-q')(q-p')]
\end{aligned} \tag{3.6.20}$$

where $p, q, r, p', q', r' \in [0,1]$.

To simplify our expression, we introduce f_1:

$$f_1 = \frac{(q'-q)(q-p')}{(p-q)(p+q-p'-q')},\ p \neq q,\ p+q \neq p'+q' \tag{3.6.21}$$

If the difference between the Gini Index functions corresponding to the tests T, T' is positive, then the favorite test for the Gini Index criterion is T, and if this difference is negative, then the favorite test is T'. The same holds for the Information Gain functions.

The difference corresponding to the Information Gain functions is expressed as follows:

$$\begin{aligned}
\text{gain}(T) - \text{gain}(T') &= r[p\log_2(p) + (1-p)\log_2(1-p)] + (1-r)[q\log_2(q) + \\
&\quad + (1-q)\log_2(1-q)] - r'[p'\log_2(p') + (1-p')\log_2(1-p')] + \\
&\quad + (1-r')[q'\log_2(q') + (1-q')\log_2(1-q')]
\end{aligned} \tag{3.6.22}$$

where $p, q, p', q' \in (0,1)$ and $r, r' \in [0,1]$.

To simplify this expression, we will use the function $f(x)$ to substitute $x\log_2(x) + (1-x)\log_2(1-x)$. We have $f : (0,1) \to [-1,0)$. Its derivative is negative on the interval $\left(0, \frac{1}{2}\right]$ and positive on the interval $\left[\frac{1}{2}, 1\right)$. Its second derivative is positive on $(0,1)$. Thus, this function is monotonically decreasing on $\left(0, \frac{1}{2}\right]$ and monotonically increasing on $\left[\frac{1}{2}, 1\right)$. It is a strictly convex function.

Using the function f defined before, the difference between the Information Gain functions corresponding to the tests T, T' can be rewritten as:

$$\text{gain}(T) - \text{gain}(T') = r(f(p) - f(q)) - r'(f(p') - f(q')) + f(q) - f(q') \tag{3.6.23}$$

Using (3.6.12), we obtain:

$$\begin{aligned}
&\text{gain}(T) - \text{gain}(T') = r\left(f(p) - f(q)\right) - \frac{r(p-q) + q - q'}{p' - q'}\left(f(p') - f(q')\right) + f(q) - f(q') \\
&= r\left[\left(f(p) - f(q)\right) - \frac{p-q}{p'-q'}\left(f(p') - f(q')\right)\right] - \frac{q-q'}{p'-q'}\left(f(p') - f(q')\right) + f(q) - f(q') \\
&= \frac{r}{p'-q'}\left[\left(f(p) - f(q)\right)(p'-q') - \left(f(p') - f(q')\right)(p-q)\right] - \\
&\quad - \frac{1}{p'-q'}\left[(q-q')\cdot\left(f(p') - f(q')\right) - \left(f(q) - f(q')\right)(p'-q')\right] \\
&= \frac{1}{p'-q'}\{r\left[\left(f(p) - f(q)\right)(p'-q') - \left(f(p') - f(q')\right)(q-p)\right] + \\
&\quad + \left(f(q) - f(q')\right)\cdot(p'-q') - \left(f(p') - f(q')\right)(q-q')\}
\end{aligned} \tag{3.6.24}$$

Now we apply the Lagrange theorem to the function f on the intervals: $[p,q]$, $[p',q']$, and $[q,q']$. The function f is continuous on $[p,q]$, its derivative f' exists and it is finite on $[p,q]$, so by the Lagrange theorem we have:

$$\exists\, x_1 \in (p,q),\ f'(x_1) = \frac{f(p) - f(q)}{p - q} \tag{3.6.25}$$

For $[p',q']$ the theorem's conditions are also satisfied and therefore:

$$\exists\, x_2 \in (p',q'),\ f'(x_2) = \frac{f(p') - f(q')}{p' - q'} \tag{3.6.26}$$

and similarly for $[q,q']$ we have:

$$\exists\, x_3 \in (q,q'),\ f'(x_3) = \frac{f(q) - f(q')}{q - q'} \tag{3.6.27}$$

We express the Information Gain difference as:

$$\begin{aligned}\text{gain}(T) - \text{gain}(T') &= \frac{1}{p'-q'}\{r[(f'(x_1)(p-q)(p'-q') - f'(x_2)(p'-q')(q-q')] + \\ &\quad + f'(x_3)(q-q')(p'-q') - f'(x_2)(p'-q')(q-q')\} \\ &= r[f'(x_1)(p-q) - f'(x_2)(p-q)] + f'(x_3)(q-q') - f'(x_2)(q-q') \\ &= r(p-q)(f'(x_1) - f'(x_2)) + (q-q')(f'(x_3) - f'(x_2)) \\ &= rE_1 + E_2 \qquad (3.6.28)\end{aligned}$$

where $p' \neq q'$, $E_1 = (p-q)(f'(x_1) - f'(x_2))$, and $E_2 = (q-q')(f'(x_3) - f'(x_2))$.

We will establish the sign of this difference taking into account the conditions (3.6.15), (3.6.16), (3.6.17), (3.6.18), $p \neq q$, $p' \neq q'$, and $q \neq q'$.

We denote by f_2 the ratio:

$$f_2 = \frac{-E_2}{E_1} = \frac{(q-q')(f'(x_2) - f'(x_3))}{(q-p)(f'(x_2) - f'(x_1))} \qquad (3.6.29)$$

The following proposition is used in our analysis to establish the order of the points x_1, x_2, x_3.

Proposition 3.6.1. *If f is a strictly convex function defined on $(0,1)$ and $0 < a < b < c < 1$ we have:*

$$\frac{f(b)-f(a)}{b-a} < \frac{f(c)-f(a)}{c-a} < \frac{f(c)-f(b)}{c-b} \qquad (3.6.30)$$

Proof:

$a < b < c \Rightarrow b = \lambda a + (1-\lambda)c$, with $\lambda \in (0,1)$. $f(b) = f(\lambda a + (1-\lambda)c) < \lambda f(a) + (1-\lambda)f(c)$ by the strictly convexity of f. We have $f(b) - f(a) < (1-\lambda)(f(c) - f(a))$. So $\frac{f(b)-f(a)}{b-a} < \frac{(1-\lambda)(f(c)-f(a))}{(1-\lambda)(c-a)} = \frac{f(c)-f(a)}{c-a}$ $(*)$.

We have $f(b) - f(c) = f(\lambda a + (1-\lambda)c) - f(c) < \lambda(f(a) - f(c))$. So $\frac{f(b)-f(c)}{b-c} > \frac{\lambda(f(a)-f(c))}{\lambda(a-c)} = \frac{f(a)-f(c)}{a-c}$ $(**)$.

The proposition results from $(*)$ and $(**)$.

As $r, r' \in [0,1]$ and (3.6.12) must be satisfied, the terms $p'-q'$, $q'-q$, and $q-p$ can not be simultaneously positive or simultaneously negative, consequently, two terms are negative and one is positive or one term is negative and two terms are positive. Thus, the characterization of the Gini Index and Information Gain functions will be done for the

following six cases:

$$(1)\begin{cases} p'-q'>0 \\ q-p>0 \\ q'-q<0 \end{cases} \quad (2)\begin{cases} p'-q'>0 \\ q-p<0 \\ q'-q>0 \end{cases} \quad (3)\begin{cases} p'-q'<0 \\ q-p>0 \\ q'-q>0 \end{cases}$$

$$(4)\begin{cases} p'-q'<0 \\ q-p<0 \\ q'-q>0 \end{cases} \quad (5)\begin{cases} p'-q'<0 \\ q-p>0 \\ q'-q<0 \end{cases} \quad (6)\begin{cases} p'-q'>0 \\ q-p<0 \\ q'-q<0 \end{cases} \qquad (3.6.31)$$

3.7 The Intervals of Coincidence/Non-coincidence of the Gini Index and Information Gain Criteria

In this section, we present the intervals of coincidence/non-coincidence in the choice of the split attribute for the Gini Index and Information Gain criteria. The sign of the differences of the Gini Index functions corresponding to two tests T, T' and of the Information Gain functions are established for the six situations in (3.6.31). If the sign of the difference of the Gini Index functions $\text{gini}(T) - \text{gini}(T')$ is the same as the sign of the difference of the Information Gain functions $\text{gain}(T) - \text{gain}(T')$, then the two split criteria select the same attribute to split on, otherwise the two split functions select different attributes to split on.

Theorem 3.7.1. *Suppose we have two available tests T, T' and our task is to determine if the test selected by the Gini Index or Information Gain criterion is the same or not. T and T' can be characterized by the parameters p, q, r and p', q', r' respectively. We determine the maximum and the minimum of the following probabilities: $\{p, q, p', q'\}$.*

(i) If $\max\{p, q, p', q'\}$ and $\min\{p, q, p', q'\}$ belong to the same test, if we obtain $\{p, q\}$, or $\{p', q'\}$ as minimum-maximum pair, then the two criteria of split will select the same test to split on.

(ii) If we obtain $\{p, p'\}$, $\{p, q'\}$, $\{q, p'\}$ or $\{q, q'\}$ as minimum-maximum pair, then there are two possible situations to distinguish. If $(f_1 - r)(f_2 - r) > 0$, then the two criteria choose the same test, and, if $(f_1 - r)(f_2 - r) < 0$, then the two criteria choose different tests.

Case 1: $p' - q' > 0$, $q - p > 0$, $q' - q < 0$.

This case can be subdivided into following subcases:

(a) $0 < p < q' < q < p' < 1$,

(b) $0 < p < q' < p' < q < 1$,

(c) $0 < q' < p < q < p' < 1$,

(d) $0 < q' < p < p' < q < 1$,

(e) $0 < q' < p' < p < q < 1$.

Case 1.(a): $0 < p < q' < q < p' < 1$

Proof:

We have to establish in this subcase the sign of the expression (3.6.20). First we show that $f_1 \in (0,1)$. We have $f_1 > 0$ as: $q' - q < 0$, $q - p' < 0$, $p - q < 0$ and $p + q - p' - q' < 0$. Using the expression of f_1 given in (3.6.21) we obtain: $f_1 - 1 = \frac{(q'-p)(p-p')}{(p-q)(p+q-p'-q')} < 0$ as $q' - p > 0$, $p - p' < 0$, $p - q < 0$, and $p + q - p' - q' < 0$.

For r and r' we must assure that (3.6.15), (3.6.16), (3.6.17), and (3.6.18) are satisfied. (3.6.17) is satisfied as: $p - q < 0$, $q - p' < 0$, $p' - q' > 0$ and $r \geq 0$. (3.6.18) is satisfied as: $p' - q' > 0$, $q' - p > 0$, $p - q < 0$ and $r' \geq 0$. But to verify that (3.6.15) and (3.6.16) are satisfied, it is necessary that $r \leq \frac{q'-q}{p-q}$ and $r' \leq \frac{q-q'}{p'-q'}$. Both ratios: $\frac{q'-q}{p-q}$, $\frac{q-q'}{p'-q'}$ are positive and smaller than 1, so we can conclude that for this case we have: $r \in \left[0, \frac{q'-q}{p-q}\right]$ and $r' \in \left[0, \frac{q-q'}{p'-q'}\right]$.

In addition we can easily show that $f_1 < \frac{q'-q}{p-q}$ as $f_1 - \frac{q'-q}{p-q} = \frac{q-q'}{p+q-p'-q'}$ and we have that $q - q' > 0$ and $p + q - p' - q' < 0$.

Knowing the position of r and f_1 relative to $\frac{q'-q}{p-q}$ we can establish the sign of the difference between gini(T) and gini(T') given by (3.6.20). For $r \in [0, f_1]$ we have gini(T) − gini$(T') \leq 0$ and for $r \in \left[f_1, \frac{q'-q}{p-q}\right]$ we have gini(T) − gini$(T') \geq 0$.

To evaluate the difference between gain(T) and gain(T') expressed in (3.6.28) we proceed in the same way. The conditions obtained for r and r' remain valid. We must find this time the position of f_2 and of r. First, we establish the order of x_1, x_2, x_3. These points can be ordered by considering all the possible permutations of them. Applying the proposition (3.6.30) to the probabilities $p < q < p'$, $p < q' < q$, $p < q' < p'$, and $q' < q < p'$ we find that $f'(x_1) < f'(x_3) < f'(x_2)$. And, using that f' is strictly monotonically increasing (its derivative, f'', is positive), we conclude that we have to analyze only the case $x_1 < x_3 < x_2$. The other cases would contradict the monotonicity of f'.

Now, it is easy to show that $E_1 \geq 0$, $E_2 \leq 0$ and $f_2 \in \left[0, \frac{q'-q}{p-q}\right)$. We have $f_2 < \frac{q'-q}{p-q}$ as using (3.6.29): $f_2 < \frac{q'-q}{p-q} \Leftrightarrow \frac{f'(x_2)-f'(x_3)}{f'(x_2)-f'(x_1)} < 1 \Leftrightarrow f'(x_3) > f'(x_1)$ which is true as demonstrated before. So, for $r \in [0, f_2]$ we have gain(T) − gain$(T') \leq 0$, and for $r \in \left[f_2, \frac{q'-q}{p-q}\right]$ we have gain(T) − gain$(T') \geq 0$.

In conclusion, for $0 < p < q' < q < p' < 1$ we have: $r \in \left[0, \frac{q'-q}{p-q}\right]$, $r' \in \left[0, \frac{q-q'}{p'-q'}\right]$ and $f_1, f_2 \in \left[0, \frac{q'-q}{p-q}\right]$. If $r \in [0, \min\{f_1, f_2\}]$, then the same test T' is selected by both split criteria. If $r \in (\min\{f_1, f_2\}, \max\{f_1, f_2\})$, then different splits are selected. If $r \in \left[\max\{f_1, f_2\}, \frac{q'-q}{p-q}\right]$, then the same test T is selected by both split criteria.

Case 1.(b): $0 < p < q' < p' < q < 1$

Proof:

To establish the position of r and r' we use the conditions (3.6.15), (3.6.16), (3.6.17), and (3.6.18) as we did before, and we obtain: $r \in \left[\frac{p'-q}{p-q}, \frac{q'-q}{p-q}\right]$ and $r' \in [0, 1]$. The expression $p + q - p' - q'$ can be negative or positive. If $p + q - p' - q' \leq 0$ then, as $r \geq 0$, $p - q < 0$, $q - q' > 0$, and, $q - p' > 0$, we have $\text{gini}(T) - \text{gini}(T') \geq 0$. If $p + q - p' - q' \geq 0$ then we have $f_1 \geq 1$. As $r \leq 1$, then we have $r \leq f_1$ and, therefore we have $\text{gini}(T) - \text{gini}(T') \geq 0$. Therefore, for $r \in \left[\frac{p'-q}{p-q}, \frac{q'-q}{p-q}\right]$ and $r' \in [0, 1]$ we have $\text{gini}(T) - \text{gini}(T') \geq 0$.

To evaluate the difference between the $\text{gain}(T)$ and $\text{gain}(T')$ we proceed in the same way. The conditions obtained for r and r' remain valid. The points x_1, x_2, x_3 will be ordered as in the previous case 1.(a). Applying the proposition (3.6.30) to the probabilities $p < q' < p'$, $p < q' < q$, $q' < p' < q$, and $p < p' < q$ and using that f' is strictly monotonically increasing, we conclude that we have only two possible cases: $x_1 < x_2 < x_3$, and $x_2 < x_1 < x_3$.

(i) In the case $x_1 < x_2 < x_3$, we have that $f'(x_1) < f'(x_2) < f'(x_3)$, therefore $E_1 \geq 0$ and $E_2 \geq 0$. Thus we have $\text{gain}(T) - \text{gain}(T') \geq 0$.

(ii) In the case $x_2 < x_1 < x_3$, we have that $f'(x_2) < f'(x_1) < f'(x_3)$, so $E_1 \leq 0$, $E_2 \geq 0$. We show that $f_2 > \frac{q'-q}{p-q}$: $f_2 > \frac{q'-q}{p-q} \Leftrightarrow \frac{f'(x_2)-f'(x_3)}{f'(x_2)-f'(x_1)} > 1 \Leftrightarrow f'(x_3) > f'(x_1)$ which is true as we are in the case $x_2 < x_1 < x_3$ and f' is strictly monotonically increasing. Therefore we have $\text{gain}(T) - \text{gain}(T') \geq 0$.

In conclusion, for $0 < p < q' < p' < q < 1$ we have: $r \in \left[\frac{p'-q}{p-q}, \frac{q'-q}{p-q}\right]$, $r' \in [0, 1]$, and the behavior of the two split functions is identical, both are choosing T as split.

Case 1.(c): $0 < q' < p < q < p' < 1$

Proof:

We have: $r \in [0,1]$ and $r' \in \left[\frac{p-q'}{p'-q'}, \frac{q-q'}{p'-q'}\right]$. The expression $p+q-p'-q'$ can be negative or positive. If $p+q-p'-q' \geq 0$ then, as $r \geq 0$, $p-q < 0$, $q-q' > 0$, and, $q-p' < 0$, we have $\text{gini}(T) - \text{gini}(T') \leq 0$. If $p+q-p'-q' \leq 0$ then we have $f_1 \geq 1$. As $r \leq 1$, then we have $r \leq f_1$ and, therefore we have $\text{gini}(T) - \text{gini}(T') \leq 0$. For $r \in [0,1]$ and $r' \in \left[\frac{p-q'}{p'-q'}, \frac{q-q'}{p'-q'}\right]$ we have $\text{gini}(T) - \text{gini}(T') \leq 0$.

Applying proposition (3.6.30) to the probabilities $q' < p < q$, $q' < p < p'$, $q' < q < p'$, and $p < q < p'$ and, using that f' is strictly monotonically increasing, we conclude that we have only the following cases to analyze: $x_3 < x_1 < x_2$ and $x_3 < x_2 < x_1$.

(i) If $x_3 < x_1 < x_2$ we have: $E_1 \geq 0$, $E_2 \leq 0$, and $f_2 > 1$, and therefore, $\text{gain}(T) - \text{gain}(T') < 0$. To demonstrate that $f_2 > 1$ is equivalent to show that $\frac{f'(x_2)-f'(x_3)}{f'(x_2)-f'(x_1)} > \frac{q-p}{q-q'}$. The left side of the inequality is greater than 1 as we have $\frac{f'(x_2)-f'(x_3)}{f'(x_2)-f'(x_1)} > 1 \Leftrightarrow f'(x_3) < f'(x_1)$. The right side of the inequality is strictly less than 1 as we have $\frac{q-p}{q-q'} < 1 \Leftrightarrow p > q'$. By combining these two observations the inequality to show becomes obviously.

(ii) For the other situation $x_3 < x_2 < x_1$ we have: $E_1 \leq 0$, $E_2 \leq 0$. Therefore $\text{gain}(T) - \text{gain}(T') \leq 0$.

In conclusion, for $0 < q' < p < q < p' < 1$ we have: $r \in [0,1]$, $r' \in \left[\frac{p-q'}{p'-q'}, \frac{q-q'}{p'-q'}\right]$, and the behavior of the two split functions is identical, both are choosing T' as split.

Case 1.(d): $0 < q' < p < p' < q < 1$

Proof:

Using the conditions (3.6.15), (3.6.16), (3.6.17), and (3.6.18) we obtain: $r \in \left[\frac{p'-q}{p-q}, 1\right]$, $r' \in \left[\frac{p-q'}{p'-q'}, 1\right]$ and $f_1 \in \left[\frac{p'-q}{p-q}, 1\right]$. If $r \in \left[\frac{p'-q}{p-q}, f_1\right]$, then we have $\text{gini}(T) - \text{gini}(T') \geq 0$. If $r \in [f_1, 1]$, then we have $\text{gini}(T) - \text{gini}(T') \leq 0$.

Applying the proposition (3.6.30) to the probabilities $q' < p < p'$, $q' < p < q$, $q' < p' < q$, and $p < p' < q$ and, using that f' is strictly monotonically increasing we conclude that we have only the case $x_2 < x_3 < x_1$ to analyze. As $x_2 < x_3 < x_1$ we have: $E_1 \leq 0$, $E_2 \geq 0$,

and $f_2 \in \left(\frac{p'-q}{p-q}, 1\right)$. For $r \in \left[\frac{p'-q}{p-q}, f_2\right]$ we have $\text{gain}(T) - \text{gain}(T') \geq 0$ and for $r \in [f_2, 1]$ we have $\text{gain}(T) - \text{gain}(T') \leq 0$.

(i) *Proof for* $f_2 > \frac{p'-q}{p-q}$: We demonstrate this inequality by equivalencies. First we use the expression of f_2 given in (3.6.29) and we obtain:

$$f_2 > \frac{p'-q}{p-q} \Leftrightarrow (f'(x_3) - f'(x_2))(q-q') > (f'(x_1) - f'(x_2))(q-p') \Leftrightarrow$$

After some simple calculations we obtain:

$$\Leftrightarrow f'(x_3)(q-q') + f'(x_2)(q'-p') + f'(x_1)(p'-q) > 0 \Leftrightarrow$$

We substitute $f'(x_1), f'(x_2), f'(x_3)$ by using (3.6.25), (3.6.26), (3.6.27):

$$\Leftrightarrow \frac{f(q)-f(q')}{q-q'}(q-q') + \frac{f(q')-f(p')}{q'-p'}(q'-p') + \frac{f(q)-f(p)}{q-p}(p'-q) > 0 \Leftrightarrow$$

$$\Leftrightarrow f(q)(p'-p) - f(p)(p'-q) - f(p')(q-p) > 0 \ (*)$$

As $p < p' < q$, $\exists\ \alpha \in (0,1)$, so that

$$p' = \alpha p + (1-\alpha)q$$

$$(*) \Leftrightarrow f(q)(1-\alpha) + f(p)\alpha > f(\alpha p + (1-\alpha)q)$$

which is true by the strict convexity of f.

(ii) *Proof for* $f_2 < 1$: As we did before, we use also here the proof by equivalencies. We substitute f_2 by its expression given in (3.6.29) and we obtain:

$$f_2 < 1 \Leftrightarrow (f'(x_2) - f'(x_3))(q-q') > (f'(x_2) - f'(x_1))(q-p) \Leftrightarrow$$

$$\Leftrightarrow f'(x_3)(q'-q) + f'(x_2)(p-q') + f'(x_1)(q-p) > 0 \Leftrightarrow$$

We substitute $f'(x_1), f'(x_2), f'(x_3)$ by their expressions given in (3.6.25), (3.6.26), and (3.6.27):

$$\Leftrightarrow \frac{f(q')-f(q)}{q'-q}(q'-q) + \frac{f(p')-f(q')}{p'-q'}(p-q') + \frac{f(q)-f(p)}{q-p}(q-p) > 0 \Leftrightarrow$$

$$\Leftrightarrow f(q')(p'-p) - f(p)(p'-q') + f(p')(p-q') > 0 \ (**)$$

As $q' < p < p'$, $\exists\ \alpha \in (0,1)$, so that:

$$p = \alpha q' + (1-\alpha)p'$$

$$(**) \Leftrightarrow f(p')(1-\alpha) + f(q')\alpha > f(\alpha q' + (1-\alpha)p')$$

which is true by the strict convexity of f.

In conclusion, for $0 < q' < p < p' < q < 1$ we have $r \in \left[\frac{p'-q}{p-q}, 1\right]$ and $r' \in \left[\frac{p-q'}{p'-q'}, 1\right]$. For $r \in \left[\frac{p'-q}{p-q}, \min\{f_1, f_2\}\right]$ the same test T is selected by both split criteria, for $r \in (\min\{f_1, f_2\}, \max\{f_1, f_2\})$ different splits are selected, and for $r \in [\max\{f_1, f_2\}, 1]$ the same test T' is selected by both split criteria.

Case 1.(e): $0 < q' < p' < p < q < 1$

Proof:

This case is dropped as it contradicts the conditions (3.6.17) and (3.6.18). As $p'-q' < 0$, (3.6.17) $\Leftrightarrow r(p-q)+q-p' \leq 0 \Leftrightarrow r \geq \frac{p'-q}{p-q}$. As the ratio $\frac{p'-q}{p-q}$ is strictly greater than 1, this implies that $r > 1$, but this is in contradiction with the fact that r represents a probability, so that r must be equal or less than 1. Therefore such a case can not be possible.

Case 2: $p' - q' > 0$, $q - p < 0$, $q' - q < 0$

This case can be subdivided into following subcases:

$$\begin{cases} (a)\ 0 < q < p < q' < p' < 1 \\ (b)\ 0 < q < q' < p < p' < 1 \\ (c)\ 0 < q < q' < p' < p < 1 \end{cases}$$

Case 2.(a): $0 < q < p < q' < p' < 1$

Proof: This case is dropped as it contradicts the condition (3.6.18).

Case 2.(b): $0 < q < q' < p < p' < 1$

Proof: We have: $r \in \left[\frac{q'-q}{p-q}, 1\right]$, $r' \in \left[0, \frac{p-q'}{p'-q'}\right]$ and $f_1 \in \left[\frac{q'-q}{p-q}, 1\right]$. If $r \in \left[\frac{q'-q}{p-q}, f_1\right]$ we have $\text{gini}(T) - \text{gini}(T') \geq 0$. If $r \in [f_1, 1]$ we have $\text{gini}(T) - \text{gini}(T') \leq 0$.

The points x_1, x_2, x_3 can be ordered in the following ways: $x_1 \leq x_3 \leq x_2$, $x_3 \leq x_1 \leq x_2$, $x_3 \leq x_2 \leq x_1$. Applying the proposition (3.6.30) to the points $q < q' < p$, $q < q' < p'$,

$q < p < p'$, and $q' < p < p'$ and, using that f' is strictly monotonically increasing increasing we conclude that we have only the case $x_3 \leq x_1 \leq x_2$ with $x_2 \neq x_3$. So we have: $E_1 \leq 0$, $E_2 > 0$, and $f_2 \in \left[\frac{q'-q}{p-q}, 1\right)$. So, if $r \in \left[\frac{q'-q}{p-q}, f_2\right]$ we have $\text{gain}(T) - \text{gain}(T') \geq 0$, and, if $r \in [f_2, 1]$ we have $\text{gain}(T) - \text{gain}(T') \leq 0$.

In conclusion, in the case $0 < q < q' < p < p' < 1$ we have: $r \in \left[\frac{q'-q}{p-q}, 1\right]$, $r' \in \left[0, \frac{p-q'}{p'-q'}\right]$, and $f_1, f_2 \in \left[\frac{q'-q}{p-q}, 1\right)$. If $r \in \left[\frac{q'-q}{p-q}, \min\{f_1, f_2\}\right]$ then the same test is chosen by both criteria. If $r \in (\min\{f_1, f_2\}, \max\{f_1, f_2\})$ then different tests are chosen by the two criteria, and if $r \in [\max\{f_1, f_2\}, 1]$ then the same test is chosen by both criteria.

Case 2.(c): $0 < q < q' < p' < p < 1$

Proof: We have: $r \in \left[\frac{q'-q}{p-q}, 1\right]$, $r' \in [0, 1]$. If $p + q - p' - q' \geq 0$, then we have $\text{gini}(T) - \text{gini}(T') \geq 0$. If $p + q - p' - q' \leq 0$ then, $f_1 \geq 1$, so $\text{gini}(T) - \text{gini}(T') \geq 0$. For $r \in \left[\frac{q'-q}{p-q}, 1\right]$ and $r' \in [0, 1]$ we have $\text{gini}(T) - \text{gini}(T') \geq 0$.

The points x_1, x_2, x_3 can be ordered in the following ways: $x_1 \leq x_3 \leq x_2$, $x_3 \leq x_1 \leq x_2$, and $x_3 \leq x_2 \leq x_1$. Applying the proposition (3.6.30) to the points $q < q' < p'$, $q < q' < p$, $q < p' < p$, and $q' < p' < p$ and, using that f' is strictly monotonically increasing, we conclude that we have to analyze only the cases $x_3 \leq x_1 \leq x_2$, with $x_2 \neq x_3$ and $x_3 < x_2 \leq x_1$. In the case $x_3 \leq x_1 \leq x_2$ with $x_2 \neq x_3$ we have: $E_1 \leq 0$, $E_2 \geq 0$, and $f_2 \geq 1$, so we have $\text{gain}(T) - \text{gain}(T') > 0$. In the case $x_3 < x_2 \leq x_1$ we have: $E_1 \geq 0$ and $E_2 > 0$, so we have $\text{gain}(T) - \text{gain}(T') > 0$.

In conclusion, in the case $0 < q < q' < p' < p < 1$ we have: $r \in \left[\frac{q'-q}{p-q}, 1\right]$ and $r' \in [0, 1]$ and the same split is selected by the two functions.

Case 3: $p' - q' < 0$, $q - p > 0$, $q' - q > 0$

This case can be subdivided into following subcases:

$$\begin{cases} (a)\ 0 < p < q < p' < q' < 1 \\ (b)\ 0 < p < p' < q < q' < 1 \\ (c)\ 0 < p' < p < q < q' < 1 \end{cases}$$

Case 3.(a): $0 < p < q < p' < q' < 1$

Proof: This case is dropped as it contradicts the condition (3.6.17).

Case 3.(b): $0 < p < p' < q < q' < 1$

Proof: We have: $r \in \left[0, \frac{p'-q}{p-q}\right]$, $r' \in \left[\frac{q-q'}{p'-q'}, 1\right]$ and $f_1 \in \left[0, \frac{p'-q}{p-q}\right]$. If $r \in [0, f_1]$ we have $\text{gini}(T) - \text{gini}(T') \leq 0$. If $r \in \left[f_1, \frac{p'-q}{p-q}\right]$ we have $\text{gini}(T) - \text{gini}(T') \geq 0$.

The points x_1, x_2, x_3 can be ordered in following ways: $x_1 \leq x_2 \leq x_3$, $x_1 \leq x_3 \leq x_2$, $x_2 \leq x_1 \leq x_3$. Applying the proposition (3.6.30)to the points $p < p' < q$, $p < p' < q'$, $p < q < q'$, and $p' < q < q'$ and, using that f' is strictly monotonically increasing we conclude that we have only the case $x_1 \leq x_2 \leq x_3$, with $x_1 \neq x_3$. In this case $x_1 \leq x_2 \leq x_3$ with $x_1 \neq x_3$ we have: $E_1 \geq 0$, $E_2 \leq 0$ and $f_2 \in \left[0, \frac{p'-q}{p-q}\right)$. For $r \in [0, f_2]$ we have $\text{gain}(T) - \text{gain}(T') \leq 0$ and for $r \in \left[f_2, \frac{p'-q}{p-q}\right]$ we have $\text{gain}(T) - \text{gain}(T') \geq 0$.

In conclusion, in the case $0 < p < p' < q < q' < 1$ we have: $r \in \left[0, \frac{p'-q}{p-q}\right]$, $r' \in \left[\frac{q-q'}{p'-q'}, 1\right]$ and $f_1, f_2 \in \left[0, \frac{p'-q}{p-q}\right)$. If $r \in [0, \min\{f_1, f_2\}]$, then the same test is chosen by both criteria. If $r \in (\min\{f_1, f_2\}, \max\{f_1, f_2\})$, then different tests are chosen by the two criteria. If $r \in \left[\max\{f_1, f_2\}, \frac{p'-q}{p-q}\right]$, then the same test is chosen by both criteria.

Case 3.(c): $0 < p' < p < q < q' < 1$

Proof: We have: $r \in [0, 1]$, $r' \in \left[\frac{q-q'}{p'-q'}, \frac{p-q'}{p'-q'}\right]$. If $p + q - p' - q' \geq 0$, then we have $\text{gini}(T) - \text{gini}(T') \leq 0$. If $p + q - p' - q' \leq 0$, then we have $f_1 \geq 1$ so $\text{gini}(T) - \text{gini}(T') \leq 0$. For $r \in [0, 1]$ and $r' \in \left[\frac{q-q'}{p'-q'}, \frac{p-q'}{p'-q'}\right]$ we have $\text{gini}(T) - \text{gini}(T') \leq 0$.

The points x_1, x_2, x_3 can be ordered in following ways: $x_1 \leq x_2 \leq x_3$, $x_1 \leq x_3 \leq x_2$, and $x_2 \leq x_1 \leq x_3$. Applying the proposition (3.6.30) to the points $p' < p < q$, $p' < p < q'$, $p' < q < q'$, and $p < q < q'$ and, using that f' is strictly monotonically increasing, we conclude that we have only the cases $x_1 \leq x_2 \leq x_3$, with $x_1 \neq x_3$ and $x_2 \leq x_1 < x_3$. In the case $x_1 \leq x_2 \leq x_3$ with $x_1 \neq x_3$ we have: $E_1 \geq 0$, $E_2 \leq 0$, and $f_2 > 1$. We have $\text{gain}(T) - \text{gain}(T') \leq 0$. In the case $x_2 \leq x_1 < x_3$ we have: $E_1 \leq 0$ and $E_2 \leq 0 \Rightarrow$ $\text{gain}(T) - \text{gain}(T') \leq 0$.

In conclusion, in the case $0 < p' < p < q < q' < 1$ we have $r \in [0, 1]$, $r' \in \left[\frac{q-q'}{p'-q'}, \frac{p-q'}{p'-q'}\right]$ and both criteria select the same test.

Case 4: $p' - q' < 0$, $q - p < 0$, $q' - q > 0$

This case can be subdivided into following subcases:

$$\begin{cases} (a)\ 0 < p' < q < p < q' < 1 \\ (b)\ 0 < p' < q < q' < p < 1 \\ (c)\ 0 < q < p' < p < q' < 1 \\ (d)\ 0 < q < p < p' < q' < 1 \\ (e)\ 0 < q < p' < q' < p < 1 \end{cases}$$

Case 4.(a): $0 < p' < q < p < q' < 1$

Proof: We have: $r \in [0,1]$ and $r' \in \left[\frac{p-q'}{p'-q'}, \frac{q-q'}{p'-q'}\right]$. If $p+q-p'-q' \leq 0$, then $\text{gini}(T) - \text{gini}(T') \leq 0$. If $p+q-p'-q' \geq 0$, then $f_1 \geq 1$ so $\text{gini}(T) - \text{gini}(T') \leq 0$. For $r \in [0,1]$ and $r' \in \left[\frac{p-q'}{p'-q'}, \frac{q-q'}{p'-q'}\right]$ we have $\text{gini}(T) - \text{gini}(T') \leq 0$.

The points x_1, x_2, x_3 can be ordered as is following: $x_1 \leq x_2 \leq x_3$, $x_1 \leq x_3 \leq x_2$, $x_2 \leq x_1 \leq x_3$, $x_2 \leq x_3 \leq x_1$, $x_3 \leq x_1 \leq x_2$, and $x_3 \leq x_2 \leq x_1$. Applying the proposition (3.6.30) to the points $p' < q < p$, $p' < q < q'$, $p' < p < q'$, and $q < p < q'$ and, using that f' is strictly monotonically increasing, we conclude that we have only the cases $x_1 \leq x_2 \leq x_3$ and $x_2 \leq x_1 \leq x_3$. In the case $x_1 \leq x_2 \leq x_3$ we have: $E_1 \leq 0$, $E_2 \leq 0$. So $\text{gain}(T) - \text{gain}(T') \leq 0$. In the case $x_2 \leq x_1 \leq x_3$ we have: $E_1 \geq 0$, $E_2 \leq 0$ and $f_2 \geq 1$. So $\text{gain}(T) - \text{gain}(T') \leq 0$.

In conclusion, in the case $0 < p' < q < p < q' < 1$ we have: $r \in [0,1]$, $r' \in \left[\frac{p-q'}{p'-q'}, \frac{q-q'}{p'-q'}\right]$, and both criteria select the same test.

Case 4.(b): $0 < p' < q < q' < p < 1$

Proof: We have: $r \in \left[0, \frac{q'-q}{p-q}\right]$, $r' \in \left[0, \frac{q-q'}{p'-q'}\right]$ and $f_1 \in \left[0, \frac{q'-q}{p-q}\right]$. If $r \in [0, f_1]$, then $\text{gini}(T) - \text{gini}(T') \leq 0$. If $r \in \left[f_1, \frac{q'-q}{p-q}\right]$ then $\text{gini}(T) - \text{gini}(T') \geq 0$.

The points x_1, x_2, x_3 can be ordered in the following ways: $x_1 \leq x_2 \leq x_3$, $x_1 \leq x_3 \leq x_2$, $x_2 \leq x_1 \leq x_3$, $x_2 \leq x_3 \leq x_1$, $x_3 \leq x_1 \leq x_2$, and $x_3 \leq x_2 \leq x_1$. Applying the proposition (3.6.30) to the points $p' < q < q'$, $p' < q < p$, $p' < q' < p$, and $q < q' < p$ and, using that f' is strictly monotonically increasing, we conclude that we have only the case $x_2 \leq x_3 \leq x_1$. We have: $E_1 \geq 0$, $E_2 \leq 0$, $f_2 \in \left[0, \frac{q'-q}{p-q}\right]$. So, if $r \in [0, f_2]$ we have $\text{gain}(T) - \text{gain}(T') \leq 0$, and if $r \in \left[f_2, \frac{q'-q}{p-q}\right]$ we have $\text{gain}(T) - \text{gain}(T') \geq 0$.

In conclusion, in the case $0 < p' < q < q' < p < 1$, we have: $r \in \left[0, \frac{q'-q}{p-q}\right]$, $r' \in \left[0, \frac{q-q'}{p'-q'}\right]$ and $f_1, f_2 \in \left[0, \frac{q'-q}{p-q}\right]$. If $r \in [0, \min\{f_1, f_2\}]$, then the same test is chosen by both criteria.

If $r \in (\min\{f_1, f_2\}, \max\{f_1, f_2\})$, then different tests are chosen by the two criteria. If $r \in \left[\max\{f_1, f_2\}, \frac{q'-q}{p-q}\right]$, then the same test is chosen by both criteria.

Case 4.(c): $0 < q < p' < p < q' < 1$

Proof: We have: $r \in \left[\frac{p'-q}{p-q}, 1\right]$, $r' \in \left[\frac{p-q'}{p'-q'}, 1\right]$, and $f_1 \in \left[\frac{p'-q}{p-q}, 1\right]$. If $r \in \left[\frac{p'-q}{p-q}, f_1\right]$ we have $\text{gini}(T) - \text{gini}(T') \geq 0$. If $r \in [f_1, 1]$ we have $\text{gini}(T) - \text{gini}(T') \leq 0$.

The points x_1, x_2, x_3 can be ordered in the following ways: $x_1 \leq x_2 \leq x_3$, $x_1 \leq x_3 \leq x_2$, $x_2 \leq x_1 \leq x_3$, $x_2 \leq x_3 \leq x_1$, $x_3 \leq x_1 \leq x_2$, and $x_3 \leq x_2 \leq x_1$. Applying the proposition (3.6.30) to the points $q < p' < p$, $q < p' < q'$, $q < p < q'$, and $p' < p < q'$ and, using that f' is strictly monotonically increasing, we conclude that we have only the case $x_1 \leq x_3 \leq x_2$. So we have: $E_1 \leq 0$, $E_2 \geq 0$, and $f_2 \in \left(\frac{p'-q}{p-q}, 1\right)$. For $r \in \left[\frac{p'-q}{p-q}, f_2\right]$ we have $\text{gain}(T) - \text{gain}(T') \geq 0$ and for $r \in [f_2, 1]$ we have $\text{gain}(T) - \text{gain}(T') \leq 0$.

In conclusion, in the case $0 < q < p' < p < q' < 1$, we have: $r \in \left[\frac{p'-q}{p-q}, 1\right]$, $r' \in \left[\frac{p-q'}{p'-q'}, 1\right]$, and $f_1, f_2 \in \left[\frac{p'-q}{p-q}, 1\right)$. If $r \in \left[\frac{p'-q}{p-q}, \min\{f_1, f_2\}\right]$, then the same test is chosen by both criteria. If $r \in (\min\{f_1, f_2\}, \max\{f_1, f_2\})$ then different tests are chosen by the two criteria. If $r \in [\max\{f_1, f_2\}, 1]$, then both criteria select the same test.

Case 4.(d): $0 < q < p < p' < q' < 1$

Proof: This case is dropped as it contradicts the conditions (3.6.17) and (3.6.18).

Case 4.(e): $0 < q < p' < q' < p < 1$

Proof: We have: $r \in \left[\frac{p'-q}{p-q}, \frac{q'-q}{p-q}\right]$ and $r' \in [0, 1]$. If $p + q - p' - q' \geq 0$, then $\text{gini}(T) - \text{gini}(T') \geq 0$. If $p+q-p'-q' \leq 0$, then $f_1 \geq 1$, so $\text{gini}(T) - \text{gini}(T') \geq 0$. For $r \in \left[\frac{p'-q}{p-q}, \frac{q'-q}{p-q}\right]$ and $r' \in [0, 1]$ we have $\text{gini}(T) - \text{gini}(T') \geq 0$.

The points x_1, x_2, x_3 can be ordered in following ways: $x_1 \leq x_2 \leq x_3$, $x_1 \leq x_3 \leq x_2$, $x_2 \leq x_1 \leq x_3$, $x_2 \leq x_3 \leq x_1$, $x_3 \leq x_1 \leq x_2$, and $x_3 \leq x_2 \leq x_1$. Applying the proposition (3.6.30) to the points $q < p' < q'$, $q < p' < p$, $q < q' < p$, and $p' < q' < p$ and, using that f' is strictly monotonically increasing, we conclude that we have only the cases $x_3 \leq x_1 \leq x_2$ and $x_3 \leq x_2 \leq x_1$. In the case $x_3 \leq x_1 \leq x_2$ we have: $E_1 \leq 0$, $E_2 \geq 0$, and $f_2 \geq \frac{q'-q}{p-q}$. So, for $r \in \left[\frac{p'-q}{p-q}, \frac{q'-q}{p-q}\right]$ we have $\text{gain}(T) - \text{gain}(T') \geq 0$. In the case $x_3 \leq x_2 \leq x_1$ we have:

$E_1 \geq 0$, $E_2 \geq 0$. So we have $\text{gain}(T) - \text{gain}(T') \geq 0$.

In conclusion, in the case $0 < q < p < p' < q' < 1$, we have: $r \in \left[\frac{p'-q}{p-q}, \frac{q'-q}{p-q}\right]$, $r' \in [0,1]$ and the two criteria select the same test.

Case 5: $p' - q' < 0$, $q - p > 0$, $q' - q < 0$

This case can be subdivided into following subcases:

$$\begin{cases} (a)\ 0 < p < p' < q' < q < 1 \\ (b)\ 0 < p' < p < q' < q < 1 \\ (c)\ 0 < p' < q' < p < q < 1 \end{cases}$$

Case 5.(a): $0 < p < p' < q' < q < 1$

Proof: We have: $r \in \left[\frac{q'-q}{p-q}, \frac{p'-q}{p-q}\right]$ and $r' \in [0,1]$. If $p+q-p'-q' \leq 0 \Rightarrow \text{gini}(T) - \text{gini}(T') \geq 0$. If $p + q - p' - q' \geq 0$ then, $f_1 \geq 1$, so $\text{gini}(T) - \text{gini}(T') \geq 0$. For $r \in \left[\frac{q'-q}{p-q}, \frac{p'-q}{p-q}\right]$ and $r' \in [0,1]$ we have $\text{gini}(T) - \text{gini}(T') \geq 0$.

The points x_1, x_2, x_3 can be ordered in following ways: $x_1 \leq x_2 \leq x_3$, $x_2 \leq x_1 \leq x_3$, $x_2 \leq x_3 \leq x_1$. Applying the proposition (3.6.30) to the points $p' < p' < q'$, $p < p' < q$, $p < q' < q$, and $p' < q' < q$ and, using that f' is strictly monotonically increasing, we conclude that we have only the cases $x_1 \leq x_2 < x_3$ and $x_2 \leq x_1 \leq x_3$ with $x_2 \neq x_3$. In the case $x_1 \leq x_2 < x_3$ we have: $E_1 \geq 0$ and $E_2 > 0$. So $\text{gain}(T) - \text{gain}(T') > 0$. In the case $x_2 \leq x_1 \leq x_3$, $x_2 \neq x_3$ we have: $E_1 \leq 0$, $E_2 \geq 0$, and $f_2 \geq \frac{p'-q}{p-q}$. So we obtain $\text{gain}(T) - \text{gain}(T') \geq 0$.

In conclusion, in the case $0 < p < p' < q' < q < 1$, we have $r \in \left[\frac{q'-q}{p-q}, \frac{p'-q}{p-q}\right]$, $r' \in [0,1]$, and both criteria select the same test.

Case 5.(b): $0 < p' < p < q' < q < 1$

Proof: We have: $r \in \left[\frac{q'-q}{p-q}, 1\right]$, $r' \in \left[0, \frac{p-q'}{p'-q'}\right]$ and $f_1 \in \left[\frac{q'-q}{p-q}, 1\right]$. If $r \in \left[\frac{q'-q}{p-q}, f_1\right]$ we have $\text{gini}(T) - \text{gini}(T') \geq 0$ and, if $r \in [f_1, 1]$ we have $\text{gini}(T) - \text{gini}(T') \leq 0$.

The points x_1, x_2, x_3 can be ordered in following ways: $x_1 \leq x_2 \leq x_3$, $x_2 \leq x_1 \leq x_3$, and $x_2 \leq x_3 \leq x_1$. Applying the proposition (3.6.30) to the points $p' < p < q'$, $p' < p < q$, $p' < q' < q$, and $p < q' < q$ and, using that f' is strictly monotonically increasing we conclude that we have only the case $x_2 \leq x_1 \leq x_3$ with $x_2 \neq x_3$. We have: $E_1 \leq 0$, $E_2 > 0$,

and $f_2 \in \left[\frac{q'-q}{p-q}, 1\right)$. We have for $r \in \left[\frac{q'-q}{p-q}, f_2\right]$ $\text{gain}(T) - \text{gain}(T') \geq 0$ and for $r \in [f_2, 1]$ $\text{gain}(T) - \text{gain}(T') \leq 0$.

In conclusion, in the case $0 < p' < p < q' < q < 1$ we have: $r \in \left[\frac{q'-q}{p-q}, 1\right]$, $r' \in \left[0, \frac{p-q'}{p'-q'}\right]$ and $f_1, f_2 \in \left[\frac{q'-q}{p-q}, 1\right)$. If $r \in \left[\frac{q'-q}{p-q}, \min\{f_1, f_2\}\right]$, then the same test is chosen by both criteria. If $r \in (\min\{f_1, f_2\}, \max\{f_1, f_2\})$, then different tests are chosen by the two criteria. If $r \in [\max\{f_1, f_2\}, 1]$, then the same test is chosen by both criteria.

Case 5.(c): $0 < p' < q' < p < q < 1$

Proof: This case is dropped as it contradicts the condition (3.6.18).

Case 6: $p' - q' > 0$, $q - p < 0$, $q' - q < 0$

This case can be subdivided into following subcases:

$$\begin{cases} (a)\ 0 < q' < p' < q < p < 1 \\ (b)\ 0 < q' < q < p' < p < 1 \\ (c)\ 0 < q' < q < p < p' < 1 \end{cases}$$

Case 6.(a): $0 < q' < p' < q < p < 1$

Proof: This case is dropped as it contradicts the condition (3.6.17).

Case 6.(b): $0 < q' < q < p' < p < 1$

Proof: We have: $r \in \left[0, \frac{p'-q}{p-q}\right]$ and $r' \in \left[0, \frac{q-q'}{p'-q'}\right]$. If $r \in [0, f_1]$ we have $\text{gini}(T) - \text{gini}(T') \leq 0$ and if $r \in \left[f_1, \frac{p'-q}{p-q}\right]$ we have $\text{gini}(T) - \text{gini}(T') \geq 0$.

The points x_1, x_2, x_3 can be ordered in following ways: $x_2 \leq x_3 \leq x_1$, $x_3 \leq x_1 \leq x_2$, and $x_3 \leq x_2 \leq x_1$. Applying the proposition (3.6.30) to the points $q' < q < p'$, $q' < q < p$, $q' < p' < p$, and $q < p' < p$ and, using that f' is strictly monotonically increasing, we conclude that we have only the case $x_3 \leq x_2 \leq x_1$ with $x_1 \neq x_3$. So we have: $E_1 \geq 0$, $E_2 \leq 0$, and $f_2 \in \left[0, \frac{p'-q}{p-q}\right)$. For $r \in [0, f_2]$ we have $\text{gain}(T) - \text{gain}(T') \leq 0$ and for $r \in \left[f_2, \frac{p'-q}{p-q}\right]$ we have $\text{gain}(T) - \text{gain}(T') \geq 0$.

In conclusion, in the case $0 < q' < q < p' < p < 1$ we have: $r \in \left[0, \frac{p'-q}{p-q}\right]$, $r' \in \left[0, \frac{q-q'}{p'-q'}\right)$ and $f_1, f_2 \in \left[0, \frac{p'-q}{p-q}\right]$. If $r \in [0, \min\{f_1, f_2\}]$, then the same test is chosen by both criteria. If $r \in (\min\{f_1, f_2\}, \max\{f_1, f_2\})$, then different tests are chosen by the two criteria. If

$r \in \left[\max\{f_1, f_2\}, \frac{p'-q}{p-q}\right]$ then the same test is selected by both criteria.

Case 6.(c): $0 < q' < q < p < p' < 1$

Proof: We have: $r \in [0,1]$ and $r' \in \left[\frac{q-q'}{p'-q'}, \frac{p-q'}{p'-q'}\right]$. If $p+q-p'-q' \leq 0$, then $\text{gini}(T) - \text{gini}(T') \leq 0$. If $p+q-p'-q' \geq 0$ then $f_1 \geq 1$ then $\text{gini}(T) - \text{gini}(T') \leq 0$. So for $r \in [0,1]$ and $r' \in \left[\frac{q-q'}{p'-q'}, \frac{p-q'}{p'-q'}\right]$ we have $\text{gini}(T) - \text{gini}(T') \leq 0$.

The points x_1, x_2, x_3 can be ordered in following ways: $x_2 \leq x_3 \leq x_1$, $x_3 \leq x_1 \leq x_2$, and $x_3 \leq x_2 \leq x_1$. Applying the proposition (3.6.30) to the points $q' < q < p$, $q' < q < p'$, $q' < p < p'$, and $q < p < p'$ and, using that f' is strictly monotonically increasing, we conclude that we have only the cases $x_3 \leq x_2 \leq x_1$ with $x_1 \neq x_3$ and $x_3 < x_1 \leq x_2$ In the case $x_3 < x_1 \leq x_2$ we have: $E_1 \leq 0$, $E_2 \leq 0 \Rightarrow \text{gain}(T) - \text{gain}(T') \leq 0$. In the case $x_3 \leq x_2 \leq x_1, x_1 \neq x_3$ we have: $E_1 \geq 0, E_2 \leq 0$ and $f_2 > 1$. We have $\text{gain}(T) - \text{gain}(T') \leq 0$.

In conclusion, in the case $0 < q' < q < p < p' < 1$ we have: $r \in [0,1]$, $r' \in \left[\frac{q-q'}{p'-q'}, \frac{p-q'}{p'-q'}\right]$, and both criteria select the same test.

The case $p = q$ and $p' = q'$

If $p = q$ and $p' = q'$ the Gini Index functions become: $\text{gini}(T) = 2x(1-x) - 2p(1-p)$ and $\text{gini}(T') = 2x(1-x) - 2p'(1-p')$. As the relation $r(p-q) + q = r'(p'-q') + q'$ must be satisfied, we obtain that $q = q'$. So, we have: $p = q = q' = p'$. This implies that: $\text{gini}(T) = \text{gini}(T')$.

If $p = q$ and $p' = q'$ the Information Gain functions become: $\text{gain}(T) = -x\log_2(x) - (1-x)\log_2(1-x) + p\log_2(p) + (1-p)\log_2(1-p)$ and $\text{gain}(T') = -x\log_2(x) - (1-x)\log_2(1-x) + p'\log_2(p') + (1-p')\log_2(1-p')$ As the relation $r(p-q) + q = r'(p'-q') + q'$ must be satisfied, we obtain that $q = q'$. So, we have: $p = q = q' = p'$. This implies that: $\text{gain}(T) = \text{gain}(T')$.

3.8 Synthesis of Results

Here we present a synthesis of the obtained results. Suppose we have two available tests T, T' and our task is to determine if the test selected by the Gini Index or Information Gain criterion is the same or not. T and T' can be characterized by the parameters p, q, r

and p', q', r' respectively. We determine the maximum and the minimum of the following probabilities: $\{p, q, p', q'\}$. If $\max\{p, q, p', q'\}$ and $\min\{p, q, p', q'\}$ belong to the same test, if we obtain $\{p, q\}$, or $\{p', q'\}$ as minimum-maximum pair, then the two criteria of split will select the same test to split on. On the contrary , if we obtain $\{p, p'\}$, $\{p, q'\}$, $\{q, p'\}$ or $\{q, q'\}$ as minimum-maximum pair, then there are two possible situations to distinguish. If $(f_1 - r)(f_2 - r) > 0$, then the two criteria choose the same test, and, if $(f_1 - r)(f_2 - r) < 0$, then the two criteria choose different tests.

The results obtained for the six cases identified can be resumed in the following way. For the case 1) we obtained two situations in which the two split criteria are selecting different tests; by symmetry we obtain for the case 4) two such situations. Cases 2) and 5) are similar (also by the symmetry) and for each of them we obtain one situation in which the selection of test is done differently by the two criteria. Finally, cases 3) and 6) are symmetric, and for each of them we obtain a situation of different selection.

3.9 Conclusions

Our approach in the comparison of split criteria is new by its analytical nature. We do not use experimental settings or outside measures to reveal the differences between the split criteria. By this formal analysis, we were able to study the behavior of the two well-known split criteria Gini Index and Information Gain. We give an exact mathematical description of the situations when they are choosing the same test to split on and when not. This allows us, without constructing decision trees, to decide for a given database if the Gini Index criterion and the Information Gain criterion select the same split attribute.

In order to compare the two split functions in a general way, we use the obtained results to compute the frequency of agreement or disagreement of the two split functions. In a sequence of tests, we considered all possible databases having two binary attributes and one binary decision attribute containing 50 up to 200 tuples. We calculated then for all sizes of databases the number of cases of disagreement. The number of cases of disagreement was never higher than 2% of all cases. This explains why most empirical studies concluded that there is no significant difference between the two criteria. Of course this does not exclude that for some specific databases there might be an important difference. In general, however,

both criteria behave more or less in the same way.

The split criteria formulas depend on the structure of the data set (see the probabilities involved), so they are tied to the size of the data set. This observation combined with the differences notified in the choice of an attribute by the two split criteria, Gini Index and Information Gain, leads us to analyze the behavior of split criteria in the relation with the size of the data set. In this thesis, we are not only interested in presenting the formal behavior of the two well-known split functions locally, but also in presenting their behavior in relation with the size of the databases from which the decision tree are inferred. The next chapter is dedicated to the analysis of the decision trees constructed using the Gini Index, Information Gain and our family of split functions in relation with the size of the training data sets.

Chapter 4

Selecting Optimal Split Functions for Large Data Sets

The previous chapter revealed the differences in the local selection of the appropriate attribute using the Gini Index or Information Gain as split criterion. Thus, the previous work represents the formal proof for the different results obtained in the empirical studies conducted by different researchers, more exactly we explained why, by using the Gini Index or Information Gain criteria, different decision trees are inferred. These two split criteria were defined in different ways (e.g. minimizing impurity of a subset, or minimizing entropy in a subset), and therefore they emphasize different properties of the inferred tree, such as size or classification accuracy. The efficiency of existing decision tree algorithms has been well established for relatively small data sets. Efficiency and scalability (we refer here to the ability to construct decision trees efficiently given large amounts of data) become issues of concern when these algorithms are applied to the mining of very large real-world databases. The differences noticed in the selection of split attribute by the different split criteria and the missing work in the perspective of large data sets conduct us to analyze their behavior in such a situation. As we are interested in the application of these split criteria in a KDD context (large sizes of data), we selected two well known split functions, namely Gini Index (CART) and Information Gain (C4.5) and introduced our own family of split functions and tested them on 9,000 data sets of different sizes (from 200 *to* 20,000 *tuples). The tests have shown that the two popular functions are very sensitive to the variation of the training set sizes and therefore the quality of the inferred trees is highly dependent on the training set*

size. At the same time however, we were able to show that the simplest members of the introduced family of split functions behave in a very predictable way and, furthermore, the created trees were superior to the trees inferred using the Gini Index or the Information Gain based on our evaluation criteria.

4.1 Introduction

Breiman's CART [7] and Quinlans's C4.5 [56] are systems created to be used in a machine learning, respectively in a statistical context. Very often, exactly the same systems are applied in a KDD context. However, this approach is not as straight forward as one might conclude on a first glance. For instance, we realized that depending on the hardware and on the operating system, C4.5 does not induce identical decision trees for the same large database. A more thorough analysis of the underling inference algorithms shows why these differences can occur. The split function applied by C4.5 is using numerical unstable algorithms. This instability is mainly related to the size of the training data set. The larger the data set the higher is the probability to get unstable behavior of the split function. An obvious alternative to C4.5's split function would be the split function used by CART, but there are many more as we presented in chapter 3. As we described, the large number of comparative studies of these functions were not really conclusive, and more importantly, their main focus was not on the behavior of these functions in a KDD environment. The size of the underlying data set was not taken into consideration.

In this chapter we will study the structure and the inference capabilities of decision trees induced from data sets of various sizes. We did not restrict ourselves to a given set of split functions, but we will analyze a whole family of split functions that ranges from Gini Index (used by CART) to Information Gain (used by C4.5). This approach is based on the observation that the Gini Index function is identical to the first term of the Taylor development of the Information Gain function. In contrast to the above-cited literature we will not only prospect the behavior of the two extremes (Information Gain and Gini Index), but also the intermediate terms. We will specially analyze their behavior as they are applied to a sequence of databases of growing size. The remainder of this chapter is structured in the following way: the next section justifies the introduction of our new

family of split functions. In the third section we introduce it. The fourth section describes the experimental setup and the fifth section presents the obtained results, and then we conclude.

4.2 Analysis of Information Gain and Gini Index

The notations and definitions related to the Gini Index and Information Gain criteria have been already introduced in the previous chapter. One should notice that in the Information Gain criterion, the logarithmic function is always applied to a probability that is in practice estimated by a frequency depending on the size of the training set or subset (see beginning of this section). For instance if a class c has only one instance, its probability is estimated by $p(c) = \frac{1}{\|\mathcal{L}\|}$. For a large data set this ratio will be very small, and therefore the value of the logarithmic function of this probability will tend to negative infinity. We would like to use a simpler function than the logarithmic function that allows us to manage exceptions occurring for small probabilities in a convenient manner. Instead of the logarithmic function we would prefer to use a polynomial function. An obvious candidate would be the Gini Index introduced previously. However, it was pointed out that the Gini Index is not so good as the Information Gain (see e.g. [28, 61]). Thus, we are looking for a polynomial function that would approximate the logarithmic function. The most straightforward way to obtain such a function, is to replace the logarithmic function by its Taylor series development. We will now present this approach, introduce a new family of split functions and then analyze and compare them.

4.3 Our New Family of Split Functions

The Taylor series development is a generic approach that allows us to approximate a n-times derivable function by a polynomial function of degree n. Any function f satisfying the above constraint will be approximated in a point x_0 in the following way:

$$f(x) = f(x_0) + \frac{f'(x_0)(x - x_0)}{1!} + \frac{f''(x_0)(x - x_0)^2}{2!} + \cdots + \frac{f^{(n)}(x_0)(x - x_0)^n}{n!} + R_n$$

For positive values, the logarithmic function can be derived infinitely often and therefore satisfies the necessary conditions to be developed in a Taylor series. For the development

of the logarithmic function in the point $x_0 = 1$ we obtain:

$$\ln(x) = 0 + \frac{\frac{1}{1}(x-1)}{1!} - \frac{\frac{1}{1^2}(x-1)^2}{2!} + \frac{\frac{2}{1^3}(x-1)^3}{3!} - \dots$$

This can be written as:

$$\ln(x) = \sum_{i \geq 1} \frac{(x-1)^i}{i} = x - 1 - \frac{1}{2}(x-1)^2 + \frac{1}{3}(x-1)^3 - \frac{1}{4}(x-1)^4 + \dots$$

We will now substitute the logarithmic functions in the Information Gain by its developments, using the first term of the development, the first two terms, the first three terms etc. In this way we construct a whole family of split functions. A particularly interesting function in our family is the first one, constructed using only the first term of the development of the logarithmic function.

$$\text{gain}(T) = -\sum_{i=1}^{k} p(c_i) \log_2(p(c_i)) + \sum_{i=1}^{n} p(t_i) \sum_{j=1}^{k} p(c_j|t_i) \log_2(p(c_j|t_i))$$

$$\approx -\frac{1}{\ln(2)} \left[\sum_{i=1}^{k} p(c_i)\,(p(c_i) - 1) + \sum_{i=1}^{n} p(t_i) \sum_{j=1}^{k} p(c_j|t_i)\,(p(c_j|t_i) - 1) \right]$$

which can be written as:

$$\frac{1}{\ln(2)} \left[1 - \sum_{i=1}^{k} (p(c_i))^2 - \sum_{i=1}^{n} p(t_i) \sum_{j=1}^{k} p(c_j|t_i)\,(1 - p(c_j|t_i)) \right].$$

This represents nothing else then the Gini Index times a constant. Now, we have defined a family of split functions with a first member equal to the Gini Index, followed by an infinity of functions converging toward the Information Gain[1]. These split functions are the new split criteria which we will use in our experiments.

4.4 Experiments

In order to compare the split functions with a minimum of bias, we built our own decision tree induction system. This system is a variant of the C4.5 system with the possibility to select a split function within the function family defined previously. This is the only parameter of the system we varied during the tests, everything else (growing, pruning etc.) remained unchanged.

[1] Even though the Taylor series development of the logarithmic function converges very quickly, for the trees constructed by the split functions this is not necessarily the case. Empirical results of the convergence can be found in section 4.5. A theoretical analysis of this behavior is outside the scope of this chapter.

More precisely, our system was parameterized in the following way: as split functions (split criteria) we used the Information Gain, the Gini Index and the first 49 functions of the previously defined family of split functions. We are only interested in the influence of the different split functions on constructing decision trees, and not in their influence on pruning trees, and therefore we did not use any pruning algorithms.

In order to conduct the tests we had to provide a large set of different databases. We were not able to find "real" data sets covering the whole spectrum of parameters we wanted to take into consideration. Therefore, we generated the necessary data sets according to our needs. The data sets are created using the same methodology employed by a large number of researchers from the machine learning community or KDD: [51], [12], [10], [54], and [33]. For the construction of the databases we used the following parameters:

size: the number of tuples per data set (200, 500, 1,000, 2000, 5000, 7500, 10,000, 15,000, 20,000)

#attributes: the number of attributes per data set (between 5 and 20)

domain size: number of possible discrete values per attribute (between 10 and 20)

#classes: number of classes the tuples are belonging to (between 2 and 12)

#unclassifiable tuples: percentage of tuples per data set that can not be correctly classified (between 1% and 20%)

size of classes: two types of class membership distributions were used 1) all classes have the same size, 2) the size of each class is selected randomly.

For each of the nine possible values for *size* we generated 100 databases by varying the other parameters randomly within the intervals predefined for each of them.

4.4.1 Test Setup

In order to analyze the behavior of the different split functions on data sets of different size, we conducted a separate set of tests for each set of the 100 databases generated per *size*. For each database we randomly sampled ten times 90% of the data as training set and we

used the remaining 10% as test data. Overall, we created one thousand training and test sets for the nine possible values of *size*. (We applied a 10-fold cross validation.)

We decided to evaluate 50 split functions (see parametrization of the system) on these test sets. Per training-test pair we generated 50 decision trees corresponding to the 50 different split functions selected above.

In order to compare the 50 different trees we used the following three criteria which we defined in the section 2.6:

1. tree size,
2. error rate for the training data,
3. error rate for the test data.

We denote the size of a database by m. So we have $m \in \{200, 500, 1000, 2000, 5000, 7500, 10000, 15000, 20000\}$, and we denote this set of sizes by S. For each possible size m, we generate 100 databases: $\forall m \in S$: d_m^i with $i \in \{1, 2, \ldots, 100\}$.

For each of the possible size and for each of the possible database we use a 10-fold cross validation, so: $\forall m \in S$ and $\forall i \in \{1, 2, \ldots, 100\}$ we construct d_m^{ik}, where $k \in \{1, 2, \ldots, 10\}$. We obtain $\forall m \in S$: 1000 d_m^{ik} training-test databases. In our study we used 50 different split functions to generate decision trees: f_j where $j \in \{1, 2, \ldots, 50\}$.

For each possible size, for each training-test database and for each split function we calculate:

1. tree size: $s_m^{ik,j}$,
2. error rate for the training data $e_m^{ik,j}$,
3. error rate for the test data $t_m^{ik,j}$

   $\forall m \in S$, $\forall i \in \{1, 2, \ldots, 100\}$, $\forall k \in \{1, 2, \ldots, 10\}$, and $\forall j \in \{1, 2, \ldots, 50\}$.

To obtain ranks of size, of error rate for the training data and of error rate for the test data for each split function, the procedure we used is described below. We present the procedure only for the ranks of size, the other ranks are calculated similarly.

- $\forall m \in S$, $\forall i \in \{1, 2, \ldots, 100\}$, and $\forall j \in \{1, 2, \ldots, 50\}$ we calculate the average tree size of the 10 sizes of the training-test databases: $s_m^{i*,j} = \frac{\sum_{k=1}^{10} s_m^{ik,j}}{10}$,

- $\forall m \in S$ and $\forall i \in \{1, 2, \ldots, 100\}$ we assigned ranks to sizes $s_m^{i*,j}$, we denote these ranks by: $r_m^{i*,j}$. These ranks of size were normalized as it follows:

  $\forall m \in S$, $\forall i \in \{1, 2, \ldots, 100\}$, and $\forall j \in \{1, 2, \ldots, 50\}$

$$\left| r_m^{i*,j} \right| = \frac{r_m^{i*,j} - \min_{j \in \{1,2,\ldots,50\}} r_m^{i*,j}}{\max_{j \in \{1,2,\ldots,50\}} r_m^{i*,j} - \min_{j \in \{1,2,\ldots,50\}} r_m^{i*,j}}$$

- In order to get one value for each split function, for each of the nine possible sizes of databases, we calculated the average rank achieved by each split function for the one hundred data sets per database size. $\forall m \in S$, and $\forall j \in \{1, 2, \ldots, 50\}$ we calculate the average rank of the normalized ranks of size:

$$\left| r_m^{**,j} \right| = \frac{\sum_{i=1}^{100} \left| r_m^{i*,j} \right|}{100}$$

In this way we obtain for each possible size $m \in S$ and for each split function f_j with $j \in \{1, 2, \ldots, 50\}$ the average rank of tree sizes. Analogously are calculated the others average ranks of error rates on training data and on test data. These average ranks are assigned to each split function as its ranks for a given database.

4.5 Results

In the previous section we presented three criteria, which now will be used to describe the behavior of the 50 split functions regarding databases of various sizes. The size of the generated decision trees is the first criterion used to present the behavior of the 50 split functions. We are specially interested in analyzing the influence of the size of the database on the created trees.

For each split function, for each of the nine possible sizes of databases, we calculated the average rank achieved by each split function for the one hundred data sets per database size. These values are represented in Figure 4.1.

The x-axis represents the Information Gain function, followed by the Taylor series developments using from 2 up to 49 terms and followed at the end by the Gini Index function.

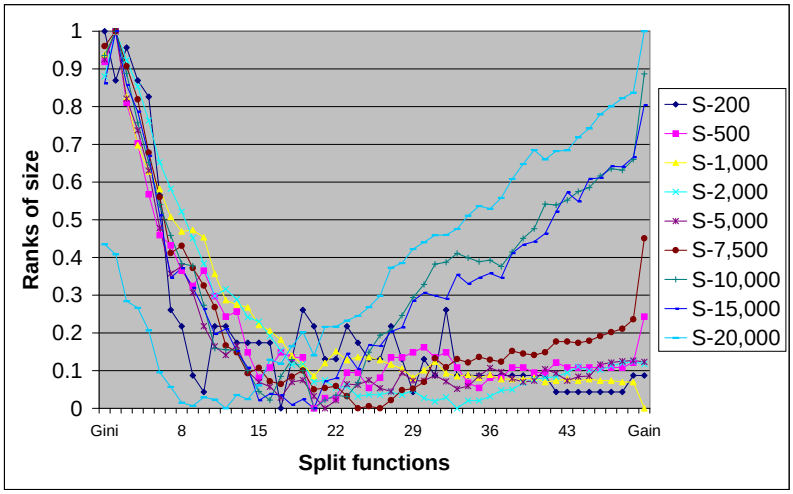

Figure 4.1: Ranks of size vs. split functions

On the y-axis we can find the average rank achieved by each split function with respect to the size of the inferred decision tree. For each database size we draw a line linking together the average ranks achieved by the 50 split functions.

First of all, we can observe that the ranks of the split functions representing approximations of the Information Gain are slowly converging toward the rank of the Information Gain. For some databases even the split functions using 50 terms of the Taylor development still create trees different from those produced by the Information Gain split function.

Secondly, we notice that the split functions based on the first few terms of the Taylor development do always create relatively large trees. The smallest trees are always obtained for split functions using between 15 and 25 terms of the Taylor development. However, as the size of the database increases, the number of terms used in the split function that achieves the optimal rank decreases. For the largest database the smallest rank with respect to the size of the inferred decision trees is achieved by a split function using 12 terms of the Taylor development, and the Information Gain function obtains the maximal rank, the Gini Index function leads to a smaller rank, but not the best one.

Furthermore, we can observe that the rank of the Information Gain increases steadily

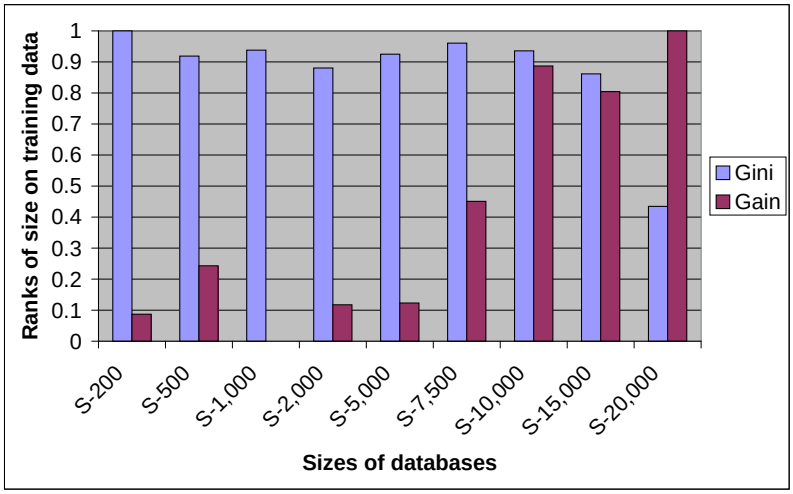

Figure 4.2: Ranks of sizes for Gini Index and Information Gain functions vs. sizes of databases

with increasing database sizes. For small databases (up to 5,000 tuples) the rank of the Information Gain can always be found in the first quarter of all ranks. Then, the rank starts to increase until it reaches finally rank=1 for a 20,000 tuples database. This can be more clearly in Figure 4.2 where we represent on the x-axis the sizes of the databases and on the y-axis we can find the average rank achieved only by the Gini Index and the Information Gain functions with respect to the size of the inferred decision trees. In other words, compared to the other split functions, the relative size of the inferred trees by Information Gain is increasing; for the largest database the Information Gain functions creates the largest tree among the trees generated by all split criteria.

In the second sequence of tests we analyze the behavior of the split functions with respect to the error rate on the training data. The results are resumed in Figure 4.3, which has the same layout as Figure 4.1, the x-axis represents the split functions, the y-axis represents this time the error-rate rank.

As for the previous criterion (size of the inferred decision tree) we can observe that the ranks of the split functions based on the Taylor developments are converging towards the

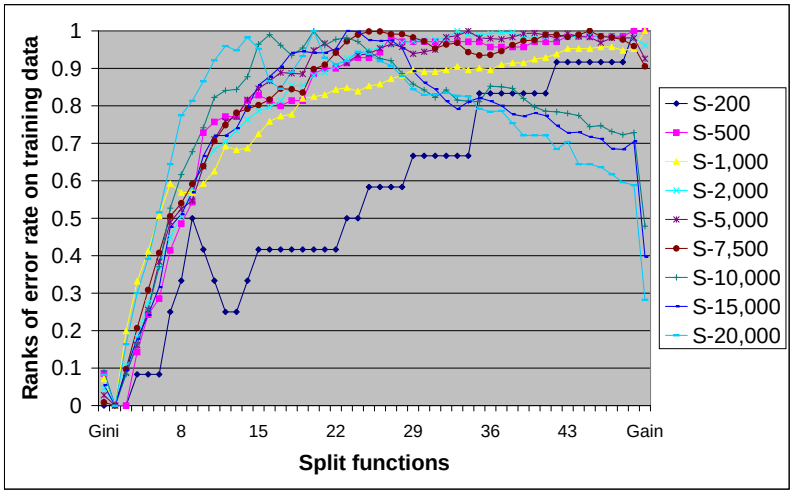

Figure 4.3: Ranks of error rate on training data vs. split functions

rank of the Information Gain function. With increasing database sizes we notice that the error-rate rank for the Information Gain is decreasing. This can be explained quite easily, by the increased relative size of the trees grown by the Information Gain (see Figure 4.1). However, for the split function based on the first, the first two and the first three terms of the Taylor development, we obtained always smaller ranks than for the Information Gain. This even holds for the cases in which the size of the trees grown by the split function based on the Taylor development was smaller than the size of the trees inferred by the Information Gain. The first three split functions based on the Taylor development achieved always the smallest ranks regardless of the database size.

In the Figure 4.4 we represent on the x-axis the possible sizes and on the y-axis we represent the average ranks achieved only by the Gini Index and Information Gain functions with respect to the error rate on the training data of the inferred decision trees.

Now, we have to analyze how the split functions behave on unseen data. Figure 4.5 pictures the error-rate rank for unseen data.

For this test set it is almost impossible to identify a general tendency. For most data sets we can not really observe any convergence. This might be related to the relatively

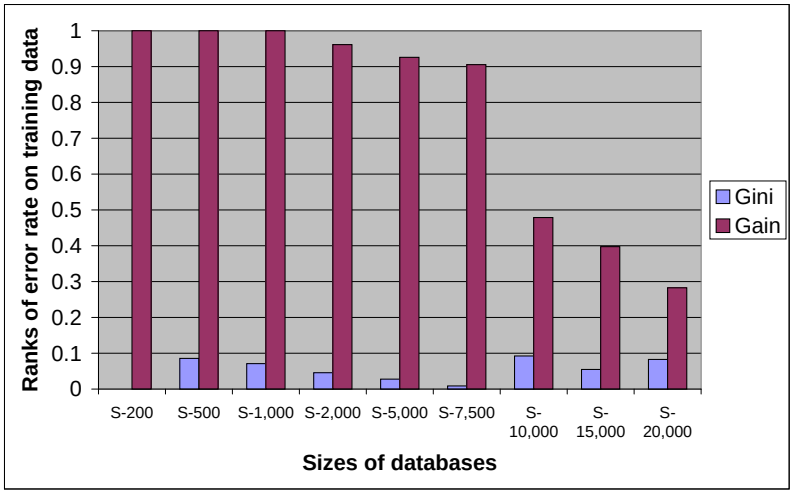

Figure 4.4: Ranks of error rate on training data for Gini Index and Information Gain functions vs. sizes of databases

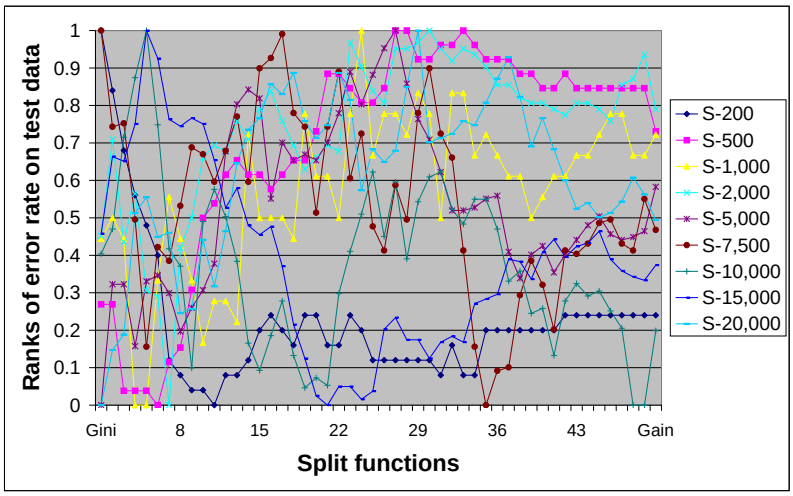

Figure 4.5: Ranks of error rate on unseen data vs. split functions

small size of the test sets used to evaluate the split functions (10% of the original database, see section 4.4.1). However, it seems important to point out that for the nine database sizes used, in seven cases the first three split functions achieved smaller ranks then the Information Gain.

4.6 Conclusion

Our main interest in this chapter was to analyze if split functions introduced in a statistical or machine learning context remain applicable in a KDD context. We introduced a new family of split functions, the two extremes being the well known Information Gain (C4.5) and Gini Index (CART). We presented a wide range of tests based on a variety of 9,000 training and test data sets. The conducted tests have revealed that the size of the databases definitely influences the behavior of the split functions. Especially the widely used Information Gain is very sensitive to the size of the training set. On the other hand we were able to show, that the three simplest split functions of the introduced family behave in a very stable way and, furthermore, the created trees were superior to the trees inferred using the Gini Index or the Information Gain based on our evaluation criteria. This is important, especially in a KDD context as the size of the training sets can vary from very small to very large.

In the actual context of KDD, characterized by an extraordinary expansion of data that are being generated and stored, another natural question which comes up is the following: How can we select smaller data sets to construct decision trees which are similar to those which we would have construct if we would have used the whole data sets? The answer to this question and an available solution is proposed in the next chapter where we describe our method of sampling.

Chapter 5

Statistical Sampling on Large Data Sets

The previous chapter was dedicated to the analysis of the behavior of the split functions when larger and larger data sets are used for the construction of decision trees. To integrate the decision tree induction in a KDD context, it is also necessary to guarantee the scalability of this technique in order to be able to induce decision trees from very large databases. A natural idea is to perform a sampling on the training database. This chapter gives a formal and an experimental analysis of the effects on a decision tree inferred from training sets created using statistical sampling. We introduce a set of sufficient conditions to assure that the attributes selected by the Gini Index or Information Gain criterion in the original and in the sampled database are the same. Based on these conditions we introduce a new method to sample training sets for decision tree induction. We conducted an exhaustive set of tests to show that the theoretically proven properties can also be validated in the experimental setting. Our sampling technique was applied to some real databases in order to evaluate the effects of sampling on the inferred decision trees. 255 decision trees were constructed using the Information Gain function. These trees were analyzed based on six evaluation criteria. We were able to show an almost perfect behavior of the proposed sampling technique in the sense, that the decision trees inferred from the whole database as well as from the sampled database are almost always identical.

5.1 Introduction

The enormity of some data sets used in recent practical applications prompts an investigation of whether such huge training sets are necessary, and how they might be reduced without sacrificing accuracy. The existing techniques to construct decision trees were not designed to be integrated in KDD processes. Mainly, the very large amounts of data, usually available for building the training set, were not considered in the design of the decision tree induction algorithms. Quite the contrary, the algorithms are very carefully designed to make an optimal use of the available data sets.

The commonly adopted solution to solve this problem is statistical sampling. This is not surprising, as for example C4.5 uses the same technique to create data samples for training and testing the decision trees. What is surprising however, is the fact, that as far as we know, it was never shown if sampling is really a valid technique for creating right sized training data sets for decision tree induction algorithms.

5.1.1 Related Work

There exists however a large body of research that deals with the question of how to iteratively set up a training set in order to induce improved decision trees. In the literature, the example-selection methods are partitioned into those that embed the selection process within the learning algorithm, those that filter examples before passing them to the induction process, and those that wrap example selection around successive calls to learning technique procedures.

Related to decision trees, the Quinlan's windowing [56] and the Cattlet's peepholing technique [9] represent well known classes of wrapper methods for example selection.

Musick et al. [41] propose a method for sampling a database in order to create a subset of the original database, which guarantees the induction of a decision tree with a predefined error rate. The empirical results presented by Musick et al. show that an appropriate sampling can lead to good performance on a wide range of inputs.

John and Langley present in [25] generic methods for a data warehouse context to create samples that are "close enough" to the original data. "Close enough" is defined as the difference of accuracies of the induced classifiers of whole data set and of the sampled

data set. The proposed approach can be applied to a naive Bayesian classifier, but it is not appropriate for more sophisticated methods such as decision tree and rule induction.

Gehrke et al.'s BOAT [20] constructed an approximate tree using a fixed-size subsample, and then refined it by scanning the full database. Provost et al. [52] studied different strategies for mining larger and larger subsamples until accuracy asymptotes.

In contrast to systems that learn in main memory by subsampling, systems like SLIQ [31] and SPRINT [60] use all the data, and concentrate on optimizing access to disk by always reading examples (more precisely, attribute lists) sequentially. Domingos proposed in [15] the VFDT system that builds decision trees using constant memory and constant time per example and using the Hoeffding bounds to guarantee that its output is asymptotically nearly identical to that of a conventional learner. It accesses data sequentially and uses subsampling to potentially require much less than one scan.

There exists also a large body of research dealing with the question of sampling for association rule induction [30, 23, 2, 70]. The proposed methods however are closely tied to the construction of association rules and cannot be applied in a straight forward way to decision tree and decision rule induction.

However, solid knowledge about the effect of sampling in the context of decision tree induction remains an open issue. We intend to propose a different sampling technique that can be justified theoretically and practically in the context of decision trees constructed from very large data sets. We offer a sampling technique that is neither difficult to apply nor expensive. The existing methods do not provide the kind of one-shot sampling approach we aim at. Unlike to the other methods, our sampling technique is applied to the data set only once at the beginning. It is not a procedure that must be repeated several times or executed for each node of the decision tree. A single application of the sampling procedure is enough to guarantee that the decision tree constructed from the smaller data set will be close to the one constructed from the whole data set in terms of structure and evaluation criteria. We will formally show how this type of data sampling for training sets affects decision tree induction algorithms.

The next chapters are structured in the following way: we will first give a formal analysis, followed by some empirical results. We will propose a new method suited to sample training

sets for decision tree induction, we will show the decision trees stability when sampling is applied to some real databases, and, finally, we will conclude.

5.2 Splitting Functions

By analyzing the introduced impurity split functions (see chapter 2), we can observe that they are independent of the size of the database. However, once we introduce in the usual way the frequency as an estimate for the conditional probability, we tie the goodness functions to the size of the database. To illustrate this, we suppose (without loss of generality) a database with n examples, with two attribute A_1, A_2 having each one two possible outcomes: 0 or 1, and, with one decision attribute D having two classes 0 or 1.

We denote by a_{ijk} the number of tuples having a value equal to i for the attribute A_1, equal to j for the attribute A_2, and equal to k for D. For example: a_{111} gives the number of tuples having $A_1 = 1$, $A_2 = 1$, and $D = 1$; a_{010} gives the number of tuples having $A_1 = 0$, $A_2 = 1$, and $D = 0$.

The a_{ijk} will have to respect the following conditions:

$$\begin{aligned} a_{1*1} + a_{1*0} + a_{0*1} + a_{0*0} &= n \\ a_{*11} + a_{*10} + a_{*01} + a_{*00} &= n \\ a_{1*1} + a_{0*1} &= a_{*11} + a_{*01} \end{aligned} \tag{5.2.1}$$

where a_{i*k} stands for the number of tuples with $A_1 = i$ and $D = k$ regardless of the value of A_2 (i.e. $a_{i*k} = a_{i1k} + a_{i0k}$).

In this context the two goodness functions we introduced are written as:

$$\text{good}_{\text{gini}}(A_1) = 1 - \left(\frac{a_{**1}}{n}\right)^2 - \left(\frac{a_{**0}}{n}\right)^2 - \frac{2}{n}\left(\frac{a_{1*1}a_{1*0}}{a_{1**}} + \frac{a_{0*1}a_{0*0}}{a_{0**}}\right) \tag{5.2.2}$$

$$\begin{aligned} \text{good}_{\text{gain}}(A_1) = {} & \frac{a_{1*1}}{n}\log_2\left(\frac{a_{1*1}}{a_{1**}}\right) + \frac{a_{1*0}}{n}\log_2\left(\frac{a_{1*0}}{a_{1**}}\right) + \frac{a_{0*1}}{n}\log_2\left(\frac{a_{0*1}}{a_{0**}}\right) + \\ & + \frac{a_{0*0}}{n}\log_2\left(\frac{a_{0*0}}{a_{0**}}\right) - \frac{a_{**1}}{n}\log_2\left(\frac{a_{**1}}{n}\right) - \frac{a_{**0}}{n}\log_2\left(\frac{a_{**0}}{n}\right) \end{aligned} \tag{5.2.3}$$

The expression for A_2 can be written analogously. The Gini Index criterion and the Information Gain criterion will select the attribute test that maximizes the respective functions

of goodness. In our context this can be expressed in the following way:

$$T = \begin{cases} A_1 & if \quad \text{good}_{\text{crit.}}(A_1) - \text{good}_{\text{crit.}}(A_2) \geq 0 \\ A_2 & if \quad \text{good}_{\text{crit.}}(A_1) - \text{good}_{\text{crit.}}(A_2) < 0 \end{cases} \tag{5.2.4}$$

For the Gini Index we obtain:

$$\text{good}_{\text{gini}}(A_1) - \text{good}_{\text{gini}}(A_2) = -\frac{2}{n}\left(\frac{a_{1*1}a_{1*0}}{a_{1**}} + \frac{a_{0*1}a_{0*0}}{a_{0**}}\right) + \frac{2}{n}\left(\frac{a_{*11}a_{*10}}{a_{*1*}} + \frac{a_{*01}a_{*00}}{a_{*0*}}\right) \tag{5.2.5}$$

For the Information Gain we obtain:

$$\begin{aligned} \text{good}_{\text{gain}}(A_1) - \text{good}_{\text{gain}}(A_2) = \frac{a_{1*1}}{n}\log_2\left(\frac{a_{1*1}}{a_{1**}}\right) + \frac{a_{1*0}}{n}\log_2\left(\frac{a_{1*0}}{a_{1**}}\right) + \\ + \frac{a_{0*1}}{n}\log_2\left(\frac{a_{0*1}}{a_{0**}}\right) + \frac{a_{0*0}}{n}\log_2\left(\frac{a_{0*0}}{a_{0**}}\right) - \frac{a_{*11}}{n}\log_2\left(\frac{a_{*11}}{a_{*1*}}\right) - \\ - \frac{a_{*10}}{n}\log_2\left(\frac{a_{*10}}{a_{*1*}}\right) - \frac{a_{*01}}{n}\log_2\left(\frac{a_{*01}}{a_{*0*}}\right) - \frac{a_{*00}}{n}\log_2\left(\frac{a_{*00}}{a_{*0*}}\right) \end{aligned} \tag{5.2.6}$$

Now we would like to formulate a set of sufficient conditions such that the size of the database can be reduced and the attribute selected according to equation (5.2.4) remains the same. We propose the following notations for the ratios which appear in (5.2.5) and (5.2.6):

$$\begin{array}{llll} \frac{a_{1*1}}{n} = c_1 & \frac{a_{*11}}{n} = c_2 & \frac{a_{1*0}}{n} = c_3 & \frac{a_{*10}}{n} = c_4 \\ \frac{a_{0*1}}{n} = c_5 & \frac{a_{*01}}{n} = c_6 & \frac{a_{0*0}}{n} = c_7 & \frac{a_{*00}}{n} = c_8 \end{array} \tag{5.2.7}$$

For two different databases with the same constants $c_1, \ldots, c_8$ we can guarantee that the same attribute will be selected to define the split function with respect to the Gini Index criterion. It is easy to show that the same set of constraints is also sufficient to assure that the attribute selected by the Information Gain criterion is the same for all databases with the same set of constants.

If we consider another database characterized by the parameters: a'_{1*1}, a'_{*11}, a'_{1*0}, a'_{*10}, a'_{0*1}, a'_{*01}, a'_{0*0}, a'_{*00}, satisfying the conditions (5.2.7) and:

$$\begin{aligned} a'_{1*1} + a'_{1*0} + a'_{0*1} + a'_{0*0} = n' \\ a'_{*11} + a'_{*10} + a'_{*01} + a'_{*00} = n' \\ a'_{1*1} + a'_{0*1} = a'_{*11} + a'_{*01} \end{aligned} \tag{5.2.8}$$

we will obtain for the Gini Index:

$$\text{good}_{\text{gini}}(A_1)' - \text{good}_{\text{gini}}(A_2)' = -\frac{2}{n'}\left(\frac{a'_{1*1}a'_{1*0}}{a'_{1**}} + \frac{a'_{0*1}a'_{0*0}}{a'_{0**}}\right) + \frac{2}{n'}\left(\frac{a'_{*11}a'_{*10}}{a'_{*1*}} + \frac{a'_{*01}a'_{*00}}{a'_{*0*}}\right) \tag{5.2.9}$$

and for the Information Gain:

$$\begin{aligned}\text{good}_{\text{gain}}(A_1)' - \text{good}_{\text{gain}}(A_2)' = \frac{a'_{1*1}}{n'}\log_2\left(\frac{a'_{1*1}}{a'_{1**}}\right) + \frac{a'_{1*0}}{n'}\log_2\left(\frac{a'_{1*0}}{a'_{1**}}\right) + \\ + \frac{a'_{0*1}}{n'}\log_2\left(\frac{a'_{0*1}}{a'_{0**}}\right) + \frac{a'_{0*0}}{n'}\log_2\left(\frac{a'_{0*0}}{a'_{0**}}\right) - \frac{a'_{*11}}{n'}\log_2\left(\frac{a'_{*11}}{a'_{*1*}}\right) - \\ - \frac{a'_{*10}}{n'}\log_2\left(\frac{a'_{*10}}{a'_{*1*}}\right) - \frac{a'_{*01}}{n'}\log_2\left(\frac{a'_{*01}}{a'_{*0*}}\right) - \frac{a'_{*00}}{n'}\log_2\left(\frac{a'_{*00}}{a'_{*0*}}\right) \qquad (5.2.10)\end{aligned}$$

We obtain that

$$\text{good}_{\text{gini}}(A_1) - \text{good}_{\text{gini}}(A_2) = \text{good}_{\text{gini}}(A_1)' - \text{good}_{\text{gini}}(A_2)'$$

$$\text{good}_{\text{gain}}(A_1) - \text{good}_{\text{gain}}(A_2) = \text{good}_{\text{gain}}(A_1)' - \text{good}_{\text{gain}}(A_2)'$$

using $a'_{1*1} = \frac{n'}{n}a_{1*1}$, $a'_{1*0} = \frac{n'}{n}a_{1*0}$, $a'_{1**} = \frac{n'}{n}a_{1*3}$, ..., and some simple calculations.

However there exists no guarantee that the Gini Index and Information Gain criterion will select the same attribute (see chapter 4). The interpretation of the conditions given in (5.2.7) is straightforward. The constant c_1 defines the proportion of tuples of the database that have the value 1 for the attribute A_1 and belong to the class 1. All other constants can be interpreted in the same way.

Based on this observation we will introduce in the next section a statistical sampling technique applicable to the decision tree induction process.

5.3 Sampling

We have introduced in equation (5.2.7) a set of sufficient conditions to assure that the test attribute selected by the Gini Index (Information Gain) criterion will be the same for all databases satisfying these constraints. If we can guarantee that the selected test attribute remains the same for different databases, then we are allowed to select a specific database (the smallest one in our case) to construct the decision tree. The question we have to answer is how to create a subset of the original database that satisfies the equations (5.2.7).

Suppose that a sampling has been performed on the database of size n. The resulting database has the dimension n' and its new parameters are: a'_{ijk}. In order to guarantee that the same splits are performed in both databases, the databases have to satisfy the following

conditions:

$$\begin{array}{llll} \frac{a_{1*1}}{n} = \frac{a'_{1*1}}{n'} & \frac{a_{*11}}{n} = \frac{a'_{*11}}{n'} & \frac{a_{1*0}}{n} = \frac{a'_{1*0}}{n'} & \frac{a_{*10}}{n} = \frac{a'_{*10}}{n'} \\ \frac{a_{0*1}}{n} = \frac{a_{0*1}}{n'} & \frac{a_{*01}}{n} = \frac{a_{*01}}{n'} & \frac{a_{0*0}}{n} = \frac{a_{0*0}}{n'} & \frac{a_{*00}}{n} = \frac{a_{*00}}{n'} \end{array} \tag{5.3.1}$$

From $\frac{a_{1*1}}{n} = \frac{a'_{1*1}}{n'}$ we obtain $\frac{a_{1*1}}{a'_{1*1}} = \frac{n}{n'} = k$. The same transformation can be applied to the remaining constraints given in equation (5.3.1). The constant k, which is the same for all eight constraints in (5.3.1) represents the reduction in size of the sampled database relative to the original database. This allows us to define a sampling procedure to reduce the size of the database, and, at the same time, to guarantee the same selection of split attributes in the sampled and in the full database. However there is one important observation to make: the values for a_{ijk} are natural numbers as they represent the numbers of tuples that belong to one of the eight specific groups. In order to obtain a database of size n/k, all the values of a_{ijk} must be multiples of k. It is of course not realistic to impose such a condition. Especially for a database with a large number of attributes or a large number of values per attribute, this condition is very hard or even impossible to satisfy. Therefore we have to introduce some approximations for the values of the a'_{ijk}.

We will now redefine the sampling procedure in the following way. First we select a value for k (the reduction in size), then we calculate the a'_{ijk} by dividing the a_{ijk} by k. Afterwards the values of a'_{ijk} are rounded to become integer values. This can be achieved in different ways, e.g. by applying the *ceil* function. The errors introduced by rounding the values of a'_{ijk} may change the decision given by equation (5.2.4). As the selection of an attribute is based on the sign of the difference of the two goodness functions of two attributes, even very small changes of these values can inverse the selection.

The a'_{ijk} are the parameters of the goodness functions. By rounding a'_{ijk} we introduce errors. The method used to estimate the error caused by this rounding in regard to the goodness function is based on the law of the propagation of maximal errors. Given $y = f(x_1, x_2, \ldots, x_n)$, if a final result y is obtained by combining a certain number of partial results x_i, then it has to be analyzed how the maximal errors of the x_i are affecting y. We suppose that the x_i are independent of each other. By developing y as a Taylor series and by assuming that the errors are sufficiently small so that the terms of higher order than

one can be neglected, the absolute error of y can be estimated by:

$$\Delta y = \sum_{i=1}^{n} \frac{\delta y}{\delta x_i} \Delta x_i$$

and the relative errors by:

$$\Delta_r y = \frac{1}{y} \sum_{i=1}^{n} \frac{\delta y}{\delta x_i} \Delta x_i = \sum_{i=1}^{n} \frac{\delta \ln y}{\delta x_i} \Delta x_i$$

Proposition 5.3.1. *The absolute errors of the $good_{gini}$ and of the $good_{gain}$ are majorized by:*

$$\left|\Delta good_{gini}(a_{1*1}, a_{0*1}, a_{1*0})\right| \leq \frac{22}{n}$$

$$\left|\Delta good_{gain}(a_{1*1}, a_{0*1}, a_{1*0})\right| \leq \frac{10}{n} \left|\log_2 \frac{1}{n}\right|$$

Proof:

We estimate the absolute error of the $\text{good}_{\text{gini}}$ function. We have $a_{1*1} + a_{0*1} + a_{1*0} + a_{0*0} = n$, and therefore $a_{0*0} = n - a_{1*1} - a_{0*1} - a_{1*0}$.

We obtain:

$$\begin{aligned} y &= \text{good}_{\text{gini}}(a_{1*1}, a_{0*1}, a_{1*0}) \\ \Delta y &= \frac{\delta y}{\delta a_{1*1}} \Delta a_{1*1} + \frac{\delta y}{\delta a_{0*1}} \Delta a_{0*1} + \frac{\delta y}{\delta a_{1*0}} \Delta a_{1*0} \end{aligned}$$

We have:

$$\begin{aligned} \frac{\delta y}{\delta a_{1*1}} &= \frac{2}{n^2}(n - 2a_{**1}) - \frac{2a_{1*0}^2}{n a_{1**}^2} - \frac{2a_{0*1}^2}{n(n - a_{1**})^2} \\ \frac{\delta y}{\delta a_{0*1}} &= \frac{2}{n^2}(n - 2a_{**1}) - \frac{2(n - a_{1**} - 2a_{0*1})}{n(n - a_{1**})} \\ \frac{\delta y}{\delta a_{1*0}} &= \frac{-2a_{1*1}^2}{n a_{1**}^2} - \frac{2a_{0*1}^2}{n(n - a_{1**})^2} \end{aligned}$$

$$|\Delta y| \leq \left|\frac{\delta y}{\delta a_{1*1}}\right| \Delta a_{1*1} + \left|\frac{\delta y}{\delta a_{0*1}}\right| \Delta a_{0*1} + \left|\frac{\delta y}{\delta a_{1*0}}\right| \Delta a_{1*0}$$

For the individual components we obtain:

$$\left|\frac{\delta y}{\delta a_{1*1}}\right| = \left|\frac{2}{n^2}(n - 2a_{**1}) - \frac{2a_{1*0}^2}{n a_{1**}^2} - \frac{2a_{0*1}^2}{n(n - a_{1**})^2}\right|$$

$$\leq \frac{2}{n^2}(|n| + 2|a_{**1}|) + \frac{2}{n}\frac{|a_{1*0}|^2}{|a_{1**}|^2} + \frac{2}{n}\frac{|a_{0*1}|^2}{|n - a_{1**}|^2} \leq \frac{2}{n^2}(n + 2n) + \frac{2}{n} + \frac{2}{n} = \frac{10}{n}.$$

Analogously:

$$\left|\frac{\delta y}{\delta a_{0*1}}\right| = \left|\frac{2}{n^2}(n - 2a_{**1}) - \frac{2(n - a_{1**} - 2a_{0*1})}{n(n - a_{1**})}\right| \leq \frac{2}{n^2}n + \frac{2}{n^2}2n + \frac{2}{n} = \frac{8}{n}$$

and,

$$\left|\frac{\delta y}{\delta a_{1*0}}\right| = \left|\frac{-2a_{1*1}^2}{na_{1**}^2} - \frac{2a_{0*1}^2}{n(n - a_{1**})^2}\right| \leq \frac{2}{n} + \frac{2}{n} = \frac{4}{n}.$$

Furthermore, we use $\Delta a_{1*1} < 1$, $\Delta a_{0*1} < 1$, and $\Delta a_{1*0} < 1$, because the error introduced using the *ceil* function is bounded by 1. And therefore:

$$|\Delta y| \leq \left|\frac{\delta y}{\delta a_{1*1}}\right| \Delta a_{1*1} + \left|\frac{\delta y}{\delta a_{0*1}}\right| \Delta a_{0*1} + \left|\frac{\delta y}{\delta a_{1*0}}\right| \Delta a_{1*0} \leq \frac{10}{n} + \frac{8}{n} + \frac{4}{n} = \frac{22}{n}$$

The estimation of the absolute error of the $y = \text{good}_{\text{gain}}(a_{1*1}, a_{0*1}, a_{1*0})$ function can be obtained in the same way. The partial derivatives of the $\text{good}_{\text{gain}}$ function are:

$$\frac{\delta y}{\delta a_{1*1}} = \frac{1}{n}\log_2\frac{a_{1*1}}{a_{1**}} - \frac{1}{n}\log_2\frac{n - a_{1*1} - a_{0*1} - a_{1*0}}{n - a_{1**}} - \frac{1}{n}\log_2\frac{a_{**1}}{n} + \frac{1}{n}\log_2\frac{n - a_{**1}}{n},$$

$$\frac{\delta y}{\delta a_{0*1}} = \frac{1}{n}\log_2\frac{a_{0*1}}{n - a_{1**}} - \frac{1}{n}\log_2\frac{n - a_{1*1} - a_{0*1} - a_{1*0}}{n - a_{1**}} - \frac{1}{n}\log_2\frac{a_{**1}}{n} + \frac{1}{n}\log_2\frac{n - a_{**1}}{n},$$

$$\frac{\delta y}{\delta a_{1*0}} = \frac{1}{n}\log_2\frac{a_{1*0}}{a_{1**}} - \frac{1}{n}\log_2\frac{n - a_{1*1} - a_{0*1} - a_{1*0}}{n - a_{1**}}.$$

We can majorize them as follows:

$$\left|\frac{\delta y}{\delta a_{1*1}}\right| = \left|\frac{1}{n}\log_2\frac{a_{1*1}}{a_{1**}} - \frac{1}{n}\log_2\frac{n - a_{1*1} - a_{0*1} - a_{1*0}}{n - a_{1**}} - \frac{1}{n}\log_2\frac{a_{**1}}{n} + \frac{1}{n}\log_2\frac{n - a_{**1}}{n}\right|$$

$$\leq \frac{1}{n}\left|\log_2\frac{1}{n}\right| + \frac{1}{n}\left|\log_2\frac{1}{n}\right| + \frac{1}{n}\left|\log_2\frac{1}{n}\right| + \frac{1}{n}\left|\log_2\frac{1}{n}\right| = \frac{4}{n}\left|\log_2\frac{1}{n}\right|$$

Analogously we obtain: $\left|\frac{\delta y}{\delta a_{0*1}}\right| \leq \frac{4}{n}\left|\log_2\frac{1}{n}\right|$ and $\left|\frac{\delta y}{\delta a_{1*0}}\right| \leq \frac{2}{n}\left|\log_2\frac{1}{n}\right|$. So the $\text{good}_{\text{gain}}$ function can be majorized:

$$|\Delta y| \leq \left|\frac{\delta y}{\delta a_{1*1}}\right| \Delta a_{1*1} + \left|\frac{\delta y}{\delta a_{0*1}}\right| \Delta a_{0*1} + \left|\frac{\delta y}{\delta a_{1*0}}\right| \Delta a_{1*0}$$

$$\leq \frac{4}{n}\left|\log_2\frac{1}{n}\right| + \frac{4}{n}\left|\log_2\frac{1}{n}\right| + \frac{2}{n}\left|\log_2\frac{1}{n}\right| = \frac{10}{n}\left|\log_2\frac{1}{n}\right|$$

It is clear that $\frac{22}{n}$ and $\frac{10}{n}\left|\log_2\frac{1}{n}\right|$ seem to be relatively large upper bounds for the error rates of functions that themselves are bounded by 0.5. However it is important to notice

that the upper bound depends on n, the size of the database. Therefore it is possible to reduce the possible error by increasing the value for n. For $n = 1000$ we know that the error is bounded by 0.022 for the Gini Index function and by 0.099 for the Information Gain function. Both functions $\frac{1}{n}$ and $\frac{1}{n} * \log_2\left(\frac{1}{n}\right)$ are converging to 0 when n tends to ∞. Based on this analysis we can make two observations.

- First, the values of the two selected split functions are converging by increasing the size of the sampled database towards the value of the respective function if applied to the full database.

- Secondly, if a split function selects a different attribute to split on for the sampled database than in the full database, then, because of the convergence described before, we know that the quality of the two attributes with respect to the goodness of split are close. As the selection of the split criterion is a local (not a global) optimization process, these small differences in the selection of the split attribute should not influence the quality of the inferred decision tree in an important way.

5.4 Experiments

In this section we present the results of a set of experiments conducted in order to show the validity of the sampling process introduced in the previous sections. As described above, we restrict ourselves to a very simple database consisting of two test attributes with two possible values for each attribute, and a decision attribute with two possible classes.

To realize a detailed analysis we generated all the possible databases of the form described above, containing 100 examples. These databases can be characterized using the eight parameters introduced in equation (5.2.1). For instance $db(20, 30, 10, 40, 25, 25, 45, 5)$ represents a database with $a_{0*0} = 20$, $a_{1*0} = 30$, etc. Applying the sampling procedure introduced previously with a sampling factor $k = 2$ we will create a new database of the form: $db'(10, 15, 5, 20, 13, 13, 23, 3)$. According to the equation (5.2.4) we find the attribute to split on. If the attributes selected in db and db' are the same, then we consider this as a "correct" decision, if the attributes differ, then we treat this database as an "exception". Using this setting we conducted a wide range of tests. In this section we restrict ourselves

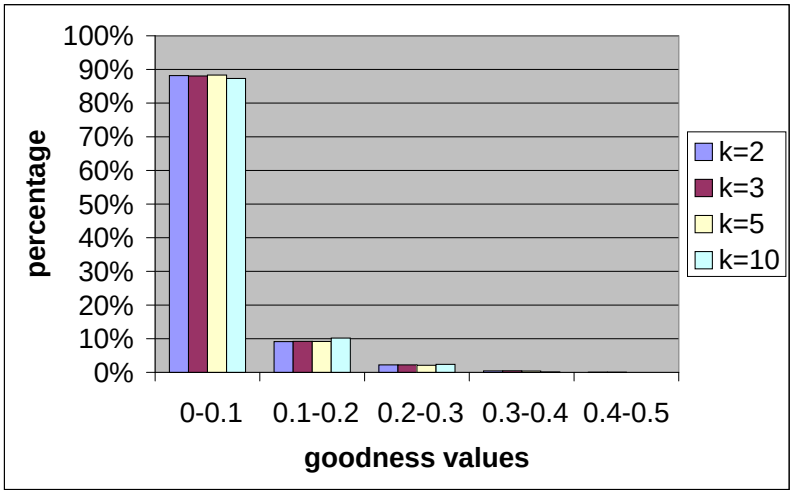

Figure 5.1: Distribution of the goodness values for all exceptions.

to two tests. All the results presented here are relative to the Gini Index function. We decided not to present the results for the Information Gain function as they are very similar to the results obtained for the Gini Index function.

The first test conducted permits to analyze the occurrences of exceptions for weak classifiers (low goodness values) as well as for strong classifiers (high goodness values). Therefore we collected all exceptions, and created a histogram of the goodness values corresponding to the attribute selected in the full database. The histogram for these values can be found in Figure 5.1. We present four series in this graph. Each series corresponds to a different sampling factor ($k = 2, 3, 5, 10$). As we can see, the large majority of exceptions can be found in the first bin, i.e. with goodness values smaller than 0.1. This is the bin that holds the category of the weakest classifiers. This is not surprising as for all possible database (with 100 examples in this test setting) most attributes deliver only weak classifiers.

The overall distribution of the goodness values is given in Figure 5.2.

If we divide the distribution of the exceptions by the distribution of the goodness values we obtain the results presented in Figure 5.3. These values can be interpreted as the "likeliness" to produce an exception for a given bin. If all values would have been equal,

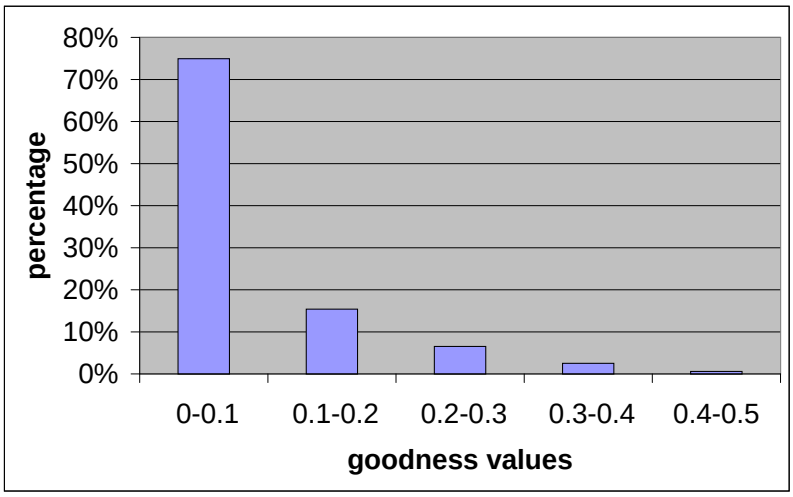

Figure 5.2: Distribution of all goodness values for the full database.

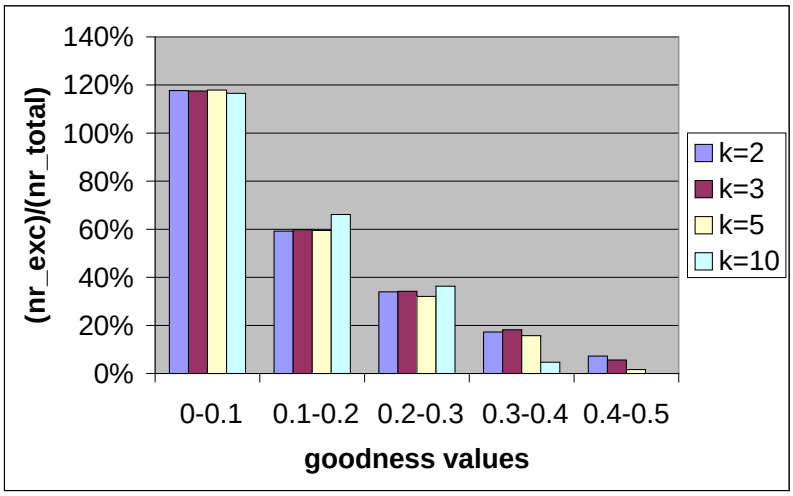

Figure 5.3: Normalized distribution of the goodness values for all exceptions.

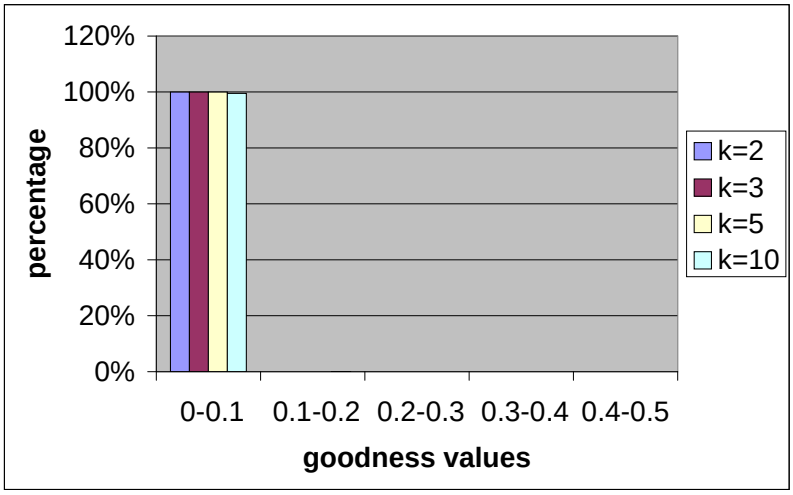

Figure 5.4: Distribution of the differences between the goodness values of the attribute selected in the full database and the attribute selected in the sampled database.

than the probability to have an exception would be the same for all bins. But this is clearly not the case for our test. The stronger a classifier becomes, the less likely it is that the proposed sampling algorithm produces an exception. This is a first important result.

The second test we conducted helps to analyze if in the case of an exception the difference between the two attributes selected is important or not with respect to their goodness values. To realize this test, we run our standard sampling procedure and calculate the difference of the goodness values relative to the full database (as the full database we take the reference database) of the two selected attributes. The results of this test are presented in Figure 5.4. We can see that almost all differences are smaller than 0.1. There are very few examples where the difference is greater than 0.1 and all of them belong to the database with the highest sampling factor $k = 10$.

We can summarize that our sampling procedure guarantees that, if there exists a strong split attribute, then this attribute will also be selected in the sampled database. In the case where not the same attributes are selected we know that the alternative attribute is in its quality/goodness still very close to the originally selected attribute.

5.5 Experiments on Real Databases

In order to study the behavior of this new sampling technique on real databases, we took five data sets from the University of California, Irvine, Repository of Machine Learning Databases (UCI) ftp://ftp.ics.uci.edu and from http://www.cs.toronto.edu, data sets arising from real-life domains. We constructed the decision trees corresponding to these data sets using the Quinlan's C4.5 system. Our procedure of sampling was applied to these data sets and we evaluated its effects on the constructed decision trees.

The data sets

We briefly describe the five data sets we used to study our sampling technique as well as any modifications that were made for the experiment.

adult: This UCI data set was donated by R. Kohavi and B. Becker. The problem is to classify the salary ("greater" or "less than 50,000") of a person using a set of features. We selected for our experiments the following 7 nominal attributes: "the education", "the marital status", "the occupation", "the relationship", "the race", "the sex" and "the native country". The data set consists in 45,222 instances. The 6 continuous attributes were eliminated.[1]

mushrooms: The mushroom records were drawn from the Audubon Society Field Guide to North American Mushrooms. The donor is Jeff Schlimmer. This data includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each of the 8,124 tuples of the data set belongs to one of the two possible classes: "edible" or "poisonous". We selected only 4 among the 22 nominally valued attributes: "the cap-shape", "the cap-surface", "the cap-color" and "the bruises" (we were obliged to proceed in this way to avoid the uniqueness of the combinations of the attributes' values if all the attributes were used).

ann-thyroid: This is an UCI data set contributed by R. Werner. The problem is to determine whether a patient referred to the clinic is hypothyroid. There are three classes: "normal", "hyperfunction", and "subnormal functioning". We selected the 15 nominal

[1] The continuous attributes present in the data sets: "adult", "ann-thyroid", and "allrep" were eliminated to avoid the uniqueness of the combinations of the values of the attributes in the database. We will not apply the grouping option existing in C4.5 to avoid the possible induced bias for our evaluations. In appendix 5.6 we present the results obtained if all the attributes are used, the continuous attributes were transformed into nominal ones.

attributes of this data set of 7,200 tuples. The 6 continuous attribute of this data set were eliminated.[1]

allrep: The thyroid disease records were supplied by the Garavan Institute and J. Ross Quinlan. There are 4 classes: "replacement therapy", "underreplacement", "overreplacement" and "negative". We selected 21 nominal attributes of the database. The 8 continuous attributes (7) were eliminated.[1]

titanic: The original source files were obtained from the data archive of the on-line Journal Of Statistics Education. The titanic data set gives the values of four categorical attributes for each of the 2021 people on board of the Titanic when it struck an iceberg and sank. The attributes are: "social class (1st, 2nd, 3rd, and crewmember)", "age (adult or child)", "sex", and "whether or not the person survived".

The System

We used C4.5, which was parameterized in the following way: as split function we used the Information Gain, the weight was set equal to 1 (to be able to compare the structure of the generated decision trees using the sampling procedure on a data set without bias: we required that any test used in the tree must have at least one outcome). In a first step we did not use the option indicating that the classifier is to be tested on unseen cases and, in a second step, we used this option.

Test Setup

We selected for k (the reduction in size of the database) the values: 2, 3, 5 and 10. For each possible combination of values of attributes of a given database, we calculate the number of tuples in the database sharing this combination. If the database has n attributes $A_1, A_2, \ldots, A_n = D$ we obtain the numbers (introduced in the previous sections) $a_{i_1,i_2\ldots i_n}$, where i_1 takes all the possible values of A_1, i_2 takes all the possible values of A_2, and i_n takes all the possible values of D. The tuples inserted in the sampled database are calculated by dividing $a_{i_1,i_2\ldots i_n}$ by k, and by rounding them to integer values as follows:

$$a_{i'_1,i'_2\ldots i'_n} = \begin{cases} \frac{a_{i_1,i_2\ldots i_n}}{k}, & if \quad \left[\frac{a_{i_1,i_2\ldots i_n}}{k}\right] = \frac{a_{i_1,i_2\ldots i_n}}{k} \\ \left\lceil \frac{a_{i_1,i_2\ldots i_n}}{k} \right\rceil, & if \quad \left[\frac{a_{i_1,i_2\ldots i_n}}{k}\right] \neq \frac{a_{i_1,i_2\ldots i_n}}{k} \end{cases}$$

To evaluate the quality of the decision trees inferred when the sampling is applied we used the following criteria presented in the section 2.6:

1. error rate on training data before pruning,
2. error rate on training data after pruning,
3. estimate of the error rate on training data after pruning,
4. error rate on test data before pruning,
5. error rate on test data after pruning,
6. estimate of the error rate on test data after pruning.

In a first step, we construct decision trees using all the examples of the data sets, without examining test sets. From each of the 5 considered databases, we constructed 4 sampled databases. The creation procedure of the sampled databases is described before. Totally, we have 25 databases to evaluate. We produce one decision tree corresponding to each of the 5 original databases and to each of the 20 sampled-databases.

In a second step, we prepared test files: we separate the examples of each database into a training set and a test set. We produce decision trees from the 5 original databases and from the sampled ones based on the training sets and we evaluate them on the prepared test sets. The training-test sets corresponding to the 25 databases taken in consideration were created in the following way: each of the five databases and each of the 20 sampled databases, was randomly sampled ten times (70% of data was taken as training set and the remaining 30% are used as test data). In this way, we generated totally 50 decision trees for each of the 5 databases. For the ten training-test data sets obtained for each full and sampled database, we calculated six average ranks corresponding to the six evaluation criteria.

Results on Real Data Sets

In the previous subsection, we introduced six criteria, which now will be used to describe the behavior of the inferred decision trees when the defined sampling procedure is applied.

In the first part, we will analyze the behavior of the first three criteria when the five original databases and the sampled ones are not splitted into training and test sets, we use all the examples to construct the decision trees. For all five databases, the behavior of the error rate on the training data, when the sampling is performed, is presented in Figure 5.5. The x-axis gives the size of the databases; the y-axis represents the error rate on training data. For the majority of the databases the error rate on the training data

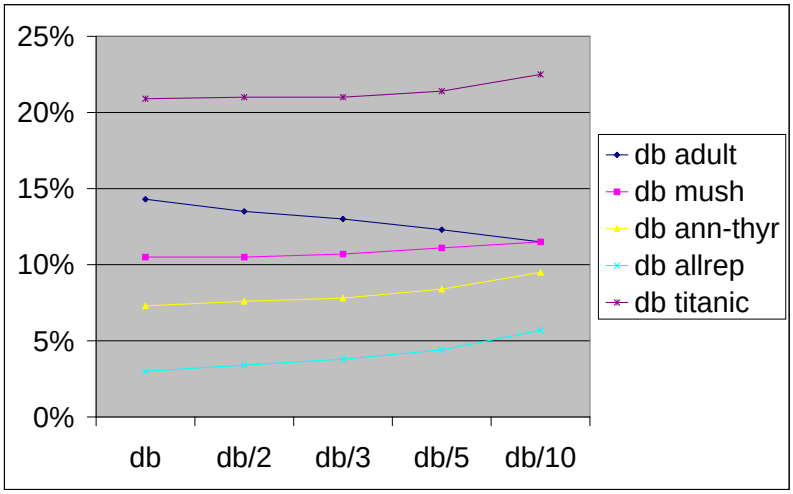

Figure 5.5: The behavior of the error rates before pruning corresponding to the whole database and to the sampled ones.

remains relatively stable when a 2, 3-sampling is performed. Beginning with a 5-sampling, the error rate increases. A different situation is observed for the largest data set considered, the "adult" one, for which the error rate on the training data decreases by sampling. The fact that the error rate on the training data for the majority of the sampled databases increases can be explained in the following way: the distribution of the combinations of all the values of the attributes includes a high numbers of counts which cannot be exactly divided by $k = 5, 10$. (Remark: $10/n \Leftrightarrow 2/n$ and $5/n$). By sampling the database, we will modify the distribution of the combinations of the values with respect to the one of the full database, so the Information Gain function may select other attributes, and the error rate of the decision tree will change. The decrease of the error rate on training data for the database "adult" is explained by the presence of the attribute "native-country", which is a 41-multi-valued one. The Information Gain criterion has a strong bias in favor of tests with many outcomes. The phenomenon of over-fitting explains the obtained result (we remember that no test set was used and no pruning was performed). The behavior of the error rate on the training data after pruning, when the sampling is performed, is presented in Figure

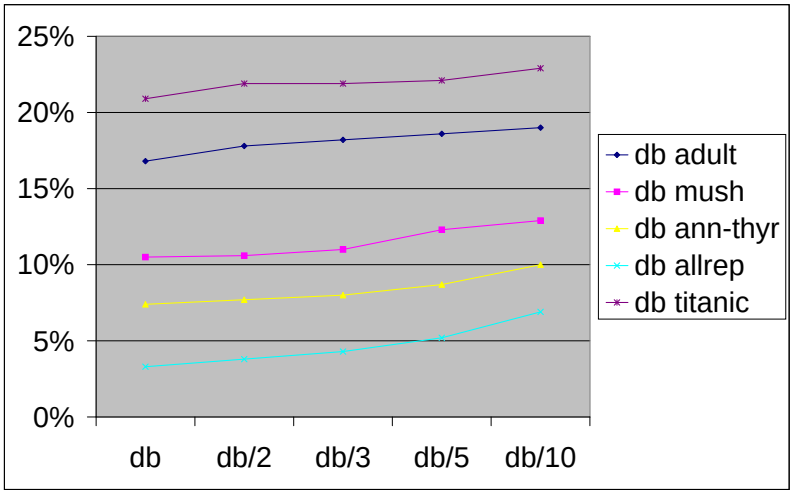

Figure 5.6: The behavior of the error rates after pruning corresponding to the whole database and to the sampled ones.

5.6. The x-axis represents the size of the databases. This time the y-axis represents the error rate on training data after pruning. The error rates after pruning have the tendency to increase slightly by sampling the databases. The estimates of the error rates when the procedure of sampling is applied have the same tendency to increase slightly. We represent them in Figure 5.7.

In the second part, we present the behavior of all the six evaluation criteria; the decision trees are constructed from the training data and tested on the test data. We remember that a 10-fold cross validation was applied for each database. In Figure 5.8 we show the behavior of all six criteria for the database "adult". The x-axis represents the size of database and the y-axis represents the average ranks with respect to the six criteria. A similar behavior as in the first part of analysis can be observed. The error rate on the training data decreases by sampling the database. We explain this by the phenomenon of over-fitting, which is eliminated by using test data and by pruning the tree. The error rate before pruning on the test data increases faster than the others, this is explained by the fact that the test data in the 2, 3, 5, 10-sampled database contains unique combinations of

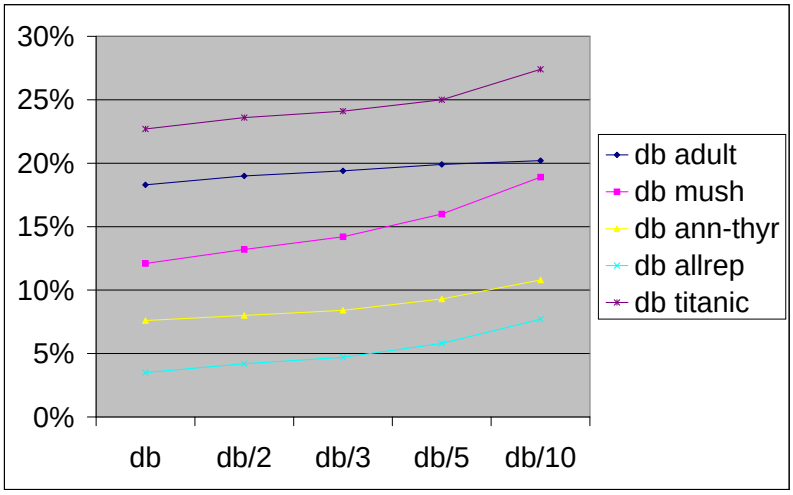

Figure 5.7: The behavior of the estimates of error rates after pruning corresponding to the whole database and to the sampled ones.

values of attributes, which are not found in 2, 3, 5, 10-training data. The other criteria have a more or less stable behavior. We will present only one another database, the "titanic", with the remark that the behavior of the criteria on the other databases is similar when our procedure of sampling is performed. In the Figure 5.9 the x-axis and the y-axis have the same interpretation as in Figure 5.8. We can notice the stability of the errors rate on the training data before pruning, after pruning, and of the estimate. The criteria which are estimated on the test data have a less stable behavior, they are increasing gradually by sampling the database. We explain this by the presence of increasingly uniquely valued tuples in the sampled database, which appear in the test set, but not in the training set. The explanation is similar to the one given before when no test set was used. More precisely by reducing the size of the database, the combinations of values of the attributes are not divisible by 2, 3, 5, 10, so we change the probabilities, which are involved in the calculation of the Information Gain function, which implies changes in the error rates of the inferred decision trees.

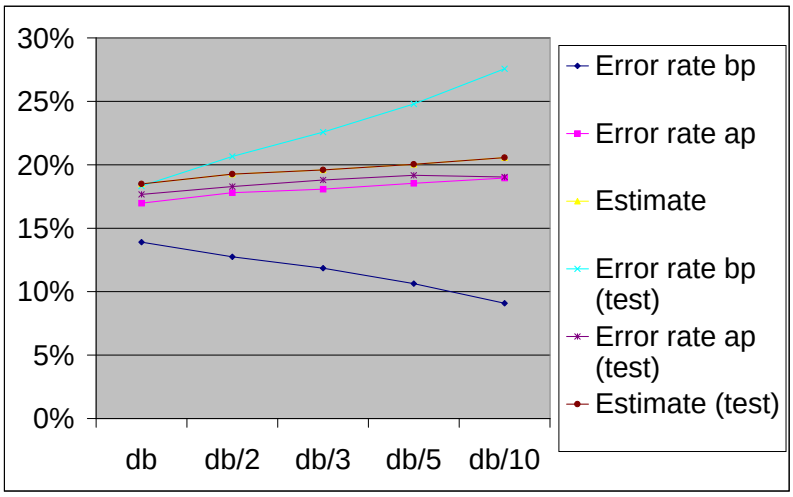

Figure 5.8: The behavior of all the criteria for adult database when sampling is performed.

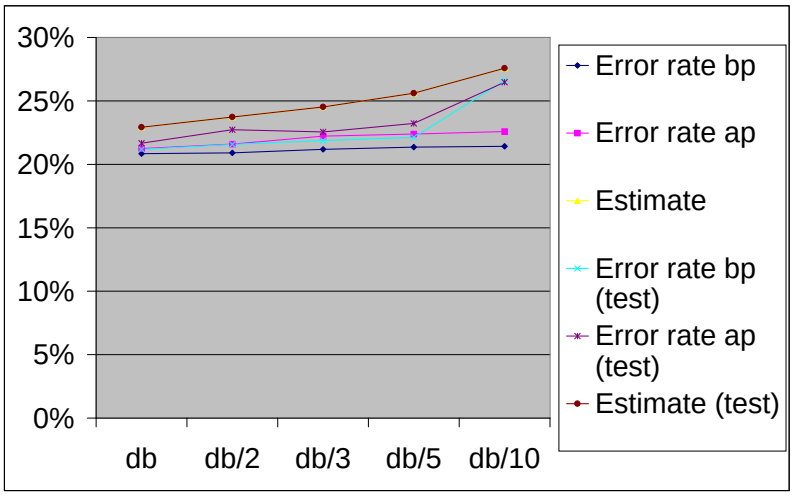

Figure 5.9: The behavior of all the criteria for titanic database when sampling is performed.

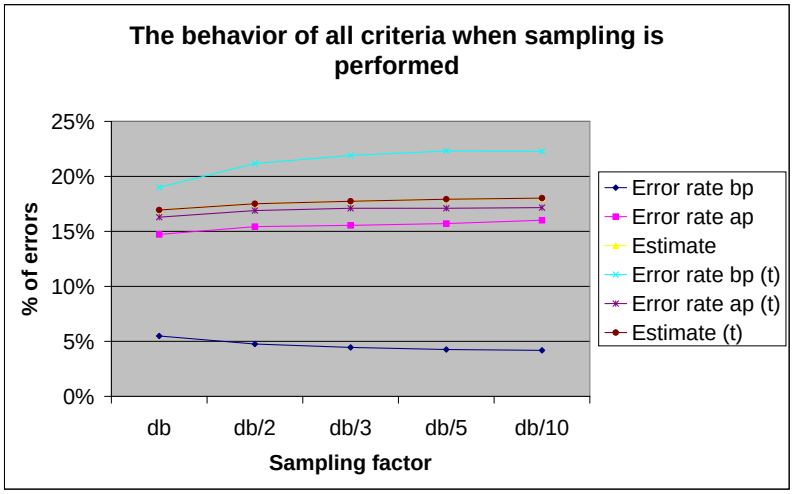

Figure 5.10: The behavior of all the criteria for adult database when sampling is performed.

5.6 Annexis: The results for the "adult" database when our sampling procedure is performed

Here we present the results we obtained when using *all the attributes* of the adult database. The continuous attributes of this database were transformed into nominal ones by using a standard procedure. Each continuous attribute was discretizated by using 10 equiwidth intervals. The width of each of 10 intervals for each attribute is calculated by adding successively the quantity $\frac{\max_{AllValuesOfAttribute} - \min_{AllValuesOfAttribute}}{10}$ to the $min_{AllValuesOfAttribute}$. The results we obtained are identically to those obtained before (see 5.10).

5.7 Conclusion

In this chapter, we presented a proportional sampling technique that allows reducing the size of the training database for a decision tree induction algorithm and guarantees that the induced trees for the full and the sampled database are very similar with respect to a given set of criteria. The procedure can be characterized using simple equations. We have formally proven that the split functions selected by the sampled database are approaching

the split functions selected by the full database with increasing size of the sampled database.

In a set of tests we have shown that the theoretical results can also be verified in practice. Tests have shown that good split attributes are also found in the sampled database. In the case when in the sampled database a different attribute is selected we know, that the selected attribute is in respect to the quality of the split still very close to the originally selected attribute.

Our technique of sampling was applied to real databases and we were able to prove the decision trees stability. This shows that the theoretically proven results hold also well in practice.

In the next chapter we will conclude the thesis by evaluating the research carried out.

Chapter 6

Conclusions

In this thesis, we explored a specific tool for data exploration and classification, namely decision trees. We started by undertaking a survey of existing work on decision trees. Then we studied in detail the Gini Index and the Information Gain criteria of split in the context of large data sets. A new family of split functions was introduced and evaluated. In the same context, a new sampling technique was defined and mathematically evaluated, before to be tested on real large data sets. We recapitulate in the next section the main questions treated in this dissertation and the given answers by it.

6.1 Summary

This thesis investigates three main questions about the induction of decision trees.

- Will the Gini Index and Information Gain split criteria select the same attribute to split on in any situation? If not, what are the conditions satisfied by the database's parameters?

✓ Our contribution is to introduce a formal methodology, which allows us to analytically compare multiple split criteria. We were able to present a formal description of how to select between split criteria for a given data set. As an illustration we applied the methodology to the two widely used split criteria: Gini Index and Information Gain.

First, the theoretical analysis of the split criteria, Gini Index and Information Gain, given in chapter 3, provides evidence that these split functions do not always select the same attribute for a given node. This explains why sometimes we obtain two

different decision trees from the same set of examples if we use two different split functions.

Secondly, the weak percentage of cases in which the two split criteria select differently (only 2%), supports the results obtained in many empirical studies which claim that there is no difference between the two split functions.

Thirdly, we define in chapter 3 the necessary and sufficient conditions to be satisfied by the database's parameters to obtain the same or a different selection of attributes of split. We can say that both criteria behave more or less in the same way.

- What will be the behavior of the Gini Index and Information Gain split functions when the size of the database from which the decision trees are inferred is increased? Are there other split functions with a more stable behavior in the case of increasing size of databases?

✓ The efficiency of existing decision tree algorithms has been well established for relatively small data sets. Efficiency and scalability become issues of concern when these algorithms are applied to the mining of very large real-world databases. The differences noticed in the selection of split attribute by the different split criteria and the missing work in the perspective of large data sets lead us to analyze their behavior in such a situation.

The conducted tests have revealed that the size of the databases definitely influences the behavior of the split functions. Especially the widely used Information Gain is very sensitive to the size of the training set. If the size of the databases increases, then the relative size of the inferred trees using the Information Gain criterion increases.

We introduced a new family of split criteria which contains also the Gini Index and Information Gain criteria. This family of split criteria is obtained by substituting the logarithmic functions in the Information Gain by its developments in Taylor series, using the first term of the development, the first two terms, etc. We were able to show, that some split functions of our family (the split functions based on the first, the first two and the first three terms of the Taylor development) behave in a very stable way and, furthermore, the created trees were superior to the trees inferred

using the Gini Index or the Information Gain based on our evaluation criteria (tree size, error rate for the training data, error rate for the test data).

- Will the trees built from large sets of examples, using the Gini Index and information Gain criteria, be equivalent to those constructed from the sampled sets of examples? What is the suitable procedure of sampling to be applied in order to keep the same structure and accuracy of the decision trees inferred from the un-sampled data sets?

✓ The enormity of some data sets used in recent practical applications prompts an investigation of whether such training sets are necessary and how they might be reduced without sacrificing accuracy.

Basing the construction of decision trees using our new technique of sampling we assure the creation of trees equivalent in structure and quality with those we would have obtained if the whole database would have been used. This technique of compaction of examples of the data set used before the application of decision tree algorithm is a useful weapon to have when attacking huge data sets.

We propose that induction from very large data sets must be done using different split criteria, by selecting the decision tree split criterion corresponding to the user's specific requirement: small size of the tree, accuracy, etc. The construction of trees from very large data sets can be made manageable with the techniques presented here. Real world applications with data sets of enormous size exist; they seem destined to grow larger. By using small samples of data (the sampling being done according to our technique) we obtain the same accuracies for the trees with those obtained if we would have used the whole set of examples to construct the decision trees.

The application of decision trees as a tool for classification has a real future. This affirmation is sustained by their advantages: their construction based on a simple and fast algorithm, the capacity to handle data with noise and missing attributes, the possibility to convert them to elementary rule sets which are easy to interpret and, as we show by this thesis, their applicability to larger and larger data sets if new techniques as those presented here are applied to ameliorate them.

Bibliography

[1] Ph.D. thesis.

[2] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo, *Fast discovery of association rules*, pp. 307–329, vol. 1 of Fayyad et al. [18], 1996.

[3] A. Babic, E. Krusinska, and J. E. Stromberg, *Extraction of diagnostic rules using recursive partitioning systems: A comparison of two approches*, Artificial Intelligence in Medicine **20** (1992), no. 5, 373–387.

[4] E. Baker and A. K. Jain, *On feature ordering in practice and some finite sample effects*, Proceedings of the Third International Joint Conference on Pattern Recognition (San Diego, CA), 1976, pp. 45–49.

[5] Moshe Ben-Bassat, *Myopic policies in sequential classification*, IEEE Transactions on Computing **27** (1978), no. 2, 170–174.

[6] Avrim L. Blum and Pat Langley, *Selection of relevant features and examples in machine learning*, Artificial Intelligence **97** (1997), 245–272.

[7] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and regression trees*, Wadsworth International Group, 1984.

[8] C. E Brodley and P. E. Utgoff, *Multivariate versus univariate decision trees*, COINS Technical Report 92-8, Department of Computer Science, University of Massachusetts, Amherst, January 1992.

[9] Jason Catlett, *Megainduction: machine learning on very large databases*, Ph.D. thesis, Basser Departement of Computer Science, The University of Sydney, 1991.

[10] J. Cheng, U. M. Fayyad, K. B. Irani, and Z. Quian, *Improved decision trees: A generalized version of id3*, Proceedings of the for the Fifth International International Conference on Machine Learning (San Mateo, California), Morgan Kaufmann, 1988.

[11] Krzysztof J. Cios, Witold Pedrycz, and Roman W. Swiniarski, *Data mining methods for knowledge discovery*, Kluwer Academic Publishers, 1998.

[12] P. Clark and T. Niblett, *The cn2 induction algorithm*, Machine Learning **3** (1989), 261–283.

[13] Lopez de Mantaras, *A distance-based attribute selection measure for decision tree induction*, Machine Learning **6** (1991), no. 1, 81–92.

[14] J.S. Deogun, V. V. Raghavan, A. Sakar, and H. Sever, *Data mining: Research trends, challenges, and applications*, Rough sets and data mining (Boston) (T. Y. Lin and N. Cercone, eds.), July 1997, pp. 9–45.

[15] Pedro Domingos and Geoff Hulten, *Mining high-speed data streams*, Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining (Boston), MA: ACM Press, 2000, pp. 71–80.

[16] B. A. Draper, C. E Brodley, and P. E. Utgoff, *Goal-directed classification using linear machine decision trees*, IEEE Transactions on Pattern Analysis and Machine Intelligence **16-9** (1994), 888–893.

[17] U. Fayyad and R. Uthurusamy (eds.), *Kdd-95 conference on knowledge discovery and data mining*, AAAI Press, 1995.

[18] Usama M. Fayyad, Gregory Piatesky-Shapiro, Padhraic Smith, and Ramasamy Uthurusamy (eds.), *Advances in knowledge discovery and data mining*, vol. 1, AAAI Press, The MIT Press, 1996.

[19] J. Gama and P. Brazdil, *Characterization of classification algorithms*, EPIA-95: Progress in Artificial Intelligence, 7th Portuguese Conference on Artificial Intelligence (C. Pinto-Ferreira and N. Mamede, eds.), Springer Verlag, 1995, pp. 189–200.

[20] J. Gehrke, V. Ganti, R. Ramakrishnan, and W.-L.Loh, *Boat: optimistic decision tree construction*, Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, 1999, pp. 169–180.

[21] A. D. Gordon, *Classification methods for the exploration analysis of multivariate data*, Chapman and Hall, London, New York, 1981.

[22] D. Heath, S. Kasif, and S. Salzberg, *Learning oblique decision trees*, IJCAI-93: Proceedings of the 13th International Joint Conference on Artificial Intelligence (Chambery, France), 1993, pp. 1002–1007.

[23] H.Toivonen, *Sampling large databases for association rules*, 22nd VLDB Conference, 1996.

[24] Laurent Hyafil and Ronald L. Rivest, *Constructing optimal binary decision trees is NP-complete*, Information Processing Letters **5** (1976), no. 1, 15–17.

[25] George H. John and P. Langley, *Static versus dynamic sampling for data mining*, Proceedings of the Second International Conference on Knowledge Discovery in Databases and Data Mining, AAAI/MIT Press, 1996.

[26] Kenneth A. Kaufman and Ryszard S. Michalski, *Learning in an inconsistent world rule selection in star/aq18*, MLI 99-1 P99-1, Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA 22030-4444, January 1999.

[27] Igor Kononenko, *On biases in estimating multi-valued attributes*, IJCAI-95: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (Montreal, Canada) (Chris Mellish, ed.), Morgan Kaufmann Publishers Inc, San Mateo, CA, August 1995, pp. 1034–1040.

[28] Tjen-Sien Lim, Wey-Yin Loh, and Yu-Shan Shih, *A comparison of prediction accuracy, complexity and training time of thirty-three old and new classification algorithms*, Machine Learning (1999).

[29] T.Y. Lin and N. Cercone (eds.), *Rough sets and data mining / analysis of imprecise data*, vol. 1, Kluwer Academic Publishers, 1997.

[30] H. Mannila, H. Toivonen, and I. Verkamo, *Efficient algorithms for discovering asociation rules*, AAAI Workshop on Knowledge Discovery in Databases (Seattle, Washington) (Usama M. Fayyad and Ramasamy Uthurusamy, eds.), July 1994, pp. 181–192.

[31] M. Mehta, A. Agrawal, and J. Rissanen, *Sliq: A fast scalable classifier for data mining*, Proceedins of the Fifth International Conference On Extanding Database Technology (Avignon, France), Springer, 1996, pp. 18–32.

[32] Usama M.Fayyad and Keki B. Irani, *The attribute specification problem in decision tree generation*, AAAI-92: Proceedings of the Tenth National Conference on Artificial Intelligence (San Jose, CA), AAAI Press/ The MIT Press, July 1992, pp. 104–110.

[33] R. S. Michalski, J. G. Carbonnel, and T. M. Mitchell, *Machine learning, an artificial intelligence approach*, vol. 1, Morgan Kaufmann, San Mateo, California, 1983.

[34] John Mingers, *Expert systems-rule induction with statistical data*, Journal of the Operational Research Society **38** (1987), no. 1, 39–47.

[35] ———, *An empirical comparison of selection measures for decision tree induction*, Machine Learning **3** (1989), 319–342.

[36] Tom M. Mitchell, *Machine learning*, machine learning international editions ed., Series in Computer Science, McGraw-Hill, 1997.

[37] Masashiro Miyakawa, *Criteria for selecting a variable in the construction of efficient decision trees*, IEEE Transactions on Computers **38** (1989), no. 1, 130–141.

[38] B. M. Moret, *Decision trees and diagrams*, Computing Surveys **14** (1982), no. 4, 593–623.

[39] Kolluru Venkata Sreerama Murthy, *On growing better decision trees from data*, Ph.D. thesis, The John Hopkins University, Baltimore, Maryland, 1995.

[40] Sreerama K. Murthy, Simon Kasif, and Steven Salzberg, *A system for induction of oblique decision trees*, Journal of artificial Intelligence research **2** (1994), 1–32.

[41] R. Musick, J. Catlett, and S. Russell, *Decision theoretic subsampling for induction on large databases*, Proceedings of the tenth International Conference on Machine Learning, 1993.

[42] G. Pagallo, *Adaptive decision tree algorithms for learning from examples*, Ph.D. thesis, University of California, Santa Cruz, 1990.

[43] G. Pagallo and D. Haussler, *Boolean feature discovery in empirical learning*, Machine Learning **5** (1990), no. 1, 71–99.

[44] Z. Pawlak, *Information systems-theoretical foundations*, Information Systems **6** (1981), 205–218.

[45] ———, *Rough sets*, International Journal of Computer and Information Sciences **11** (1982), 341–356.

[46] ———, *Rough classification*, International Journal of Man-Machine Studies **20** (1984), 469–483.

[47] ———, *Rough sets*, pp. 3–9, vol. 1 of Lin and Cercone [29], 1997.

[48] Z. Pawlak and A. Skowron, *Rough membership functions*, Rough Sets, Fuzzy Sets and Knowledge Discovery (1994), 130–135.

[49] Z. Pawlak and R. Slowinski, *Rough set approach to multi-attribute decision analysis*, European Journal of Operational research (1994), 443–459.

[50] Judea Pearl, *Probabilistic reasoning in intelligent systems / networks of plausible inference*, Representation and Reasoning, Morgan Kaufmann Publishers, Inc, San Mateo, California, 1998.

[51] Gregory Piatesky-Shapiro and Willliam J. Frawley, *Knowledge discovery in databases*, AAAI Press/ The MIT Press, 1991.

[52] F. Provost, D. Jensen, and T. Oates, *Efficient progessive sampling*, Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (San Diego, CA), ACM Press, 1999, pp. 169–180.

[53] John Ross Quinlan, *Learning efficient classification procedures and their application to chess and games*, Machine Learning: An Artificial Intelligence Approach (1983).

[54] ———, *Induction of decision trees*, Machine Learning **1** (1986), 81–106.

[55] ———, *Simplifying decision trees*, International Journal of Man-Machine Studies (1987), no. 27, 221–234.

[56] ———, *C4.5 programs for machine learning*, Morgan Kaufmann Publishers, 1993.

[57] John Ross Quinlan and Ronald L. Rivest, *Inferring decision trees using the minimum description length principle*, Information and Computation **3** (1989), no. 80, 227–248.

[58] S. R. Safavin and D. Langrebe, *A survey of decision tree classifier methodology*, IEEE Transactions on Systems, Man and Cybernetics **21** (1991), no. 3, 660–674.

[59] M. Sahami, *Learning non-linearly separable boolean functions with linear threshold unit trees and madaline-style networks*, Proceedings of the Eleventh National Conference on Artificial Inteligence (AAAI Press, ed.), 1993, pp. 335–341.

[60] J. C. Shafer, R. Agrawal, and M. Mehta, *Sprint: A scalable parallel classifier for data mining*, Proceedins of the Twenty-Second International Conference On Very Large Databases (Mumbai, India), Morgan Kaufmann, 1996, pp. 544–555.

[61] Y. Shih, *Families of splitting criteria for classification trees*, Statistics and Computing **9** (1999), 309–315.

[62] H. A. Simon and G. Lea, *Why should machines learn*, p. 28, vol. 1 of [33], 1983.

[63] P. E. Utgoff and C. E Brodley, *Linear machine decision trees*, COINS Technical Report 91-10, Department of Computer Science, University of Massachusetts, Amherst, January 1991.

[64] Ricardo Vilalta and Daniel Oblinger, *A quantification of distance-bias between evaluation metrics in classification*, Proceedings of the 17th International Conference on Machine Learning, Stanford University, 2000.

[65] C.S. Wallace, *Classification by minimum-message-length inference*, Advances in Computing and Information, 1990, pp. 72–81.

[66] Sholom M. Weiss and Casimir A. Kulikowski, *Computer systems that learn*, Morgan Kauffmann Publishers, Inc, San Francisco, California, 1991.

[67] Allan P. White and Wei Zhang Liu, *Bias in information-based measures in decision tree induction*, Machine Learning **15** (1994), no. 3, 321–329.

[68] Ian H. Witten and Eibe Frank, *Data mining practical machine learning tools and techniques with java implementations*, Morgan Kaufmann Publishers, 2000.

[69] Y.Y. Yao, S. K. M. Wong, and T. Y. Lin, *A review of rough set models*, pp. 47–77, vol. 1 of Lin and Cercone [29], 1997.

[70] Mohammed J. Zaki, Srinivasan Parthasarathy, Wei Li, and Mitsunori Ogihara, *Evaluation of sampling for data mining of association rules*, 7th International Workshop on Research Issues in Data Engineering, April 1997, pp. 42–50.