MONOGRAFIES DE L'INSTITUT D'INVESTIGACIÓ
EN INTEL·LIGÈNCIA ARTIFICIAL
Number 21

Institut d'Investigació
en Intel·ligència Artificial

Consell Superior
d'Investigacions Científiques

Monografies de l'Institut d'Investigació en Intel·ligència Artificial

Num. 1 J. Puyol, *MILORD II: A Language for Knowledge–Based Systems*

Num. 2 J. Levy, *The Calculus of Refinements, a Formal Specification Model Based on Inclusions*

Num. 3 Ll. Vila, *On Temporal Representation and Reasoning in Knowledge–Based Systems*

Num. 4 M. Domingo, *An Expert System Architecture for Identification in Biology*

Num. 5 E. Armengol, *A Framework for Integrating Learning and Problem Solving*

Num. 6 J. Ll. Arcos, *The Noos Representation Language*

Num. 7 J. Larrosa, *Algorithms and Heuristics for Total and Partial Constraint Satisfaction*

Num. 8 P. Noriega, *Agent Mediated Auctions: The Fishmarket Metaphor*

Num. 9 F. Manyà, *Proof Procedures for Multiple-Valued Propositional Logics*

Num. 10 W. M. Schorlemmer, *On Specifying and Reasoning with Special Relations*

Num. 11 M. López-Sánchez, *Approaches to Map Generation by means of Collaborative Autonomous Robots*

Num. 12 D. Robertson, *Pragmatics in the Synthesis of Logic Programs*

Num. 13 P. Faratin, *Automated Service Negotiation between Autonomous Computational Agents*

Num. 14 J. A. Rodríguez, *On the Design and Construction of Agent-mediated Electronic Institutions*

Num. 15 T. Alsinet, *Logic Programming with Fuzzy Unification and Imprecise Constants: Possibilistic Semantics and Automated Deduction*

Num. 16 A. Zapico, *On Axiomatic Foundations for Qualitative Decision Theory - A Possibilistic Approach*

Num. 17 A. Valls, *ClusDM: A multiple criteria decision method for heterogeneous data sets*

Num. 18 D. Busquets, *A Multiagent Approach to Qualitative Navigation in Robotics*

Num. 19 M. Esteva, *Electronic Institutions: from specification to development*

Num. 20 J. Sabater, *Trust and Reputation for Agent Societies*

Num. 21 J. Cerquides, *Improved Algorithms for Learning Bayesian Network Classifiers*

Num. 22 M. Villaret, *On Some Variants of Second-Order Unification*

Num. 23 M. Gómez, *Open, Reusable and Configurable Multi-Agent Systems: A Knowledge Modelling Approach*

Num. 24 S. Ramchurn, *Multi-Agent Negotiation Using Trust and Persuasion*

Improved Algorithms for Learning Bayesian Network Classifiers

Jesús Cerquides

Foreword by Ramon López de Mántaras

2005 Consell Superior d'Investigacions Científiques
Institut d'Investigació en Intel·ligència Artificial
Bellaterra, Catalonia, Spain.

Series Editor
Institut d'Investigació en Intel·ligència Artificial
Consell Superior d'Investigacions Científiques

Foreword by
Ramon López de Mántaras
Institut d'Investigació en Intel·ligència Artificial
Consell Superior d'Investigacions Científiques

Volume Author
Jesús Cerquides
Institut d'Investigació en Intel·ligència Artificial
Consell Superior d'Investigacions Científiques

Institut d'Investigació
en Intel·ligència Artificial

Consell Superior
d'Investigacions Científiques

NIPO: 653-05-060-4
ISBN: 84-00-08317-2
Dip. Legal: B-36527-2005

Ordering Information: Text orders should be addressed to the Library of the IIIA, Institut d'Investigació en Intel·ligència Artificial, Campus de la Universitat Autònoma de Barcelona, 08193 Bellaterra, Barcelona, Spain.

Contents

List of Figures

Foreword

During the last ten years, the field of Machine Learning has made impressive progress partially due to the adoption of very sophisticated mathematical machinery, specially from the fields of probability theory and statistics. One of the best examples of this progress is the subfield of Bayesian network learning. This monography describes very fine results obtained by Dr. Jesús Cerquides during several years of research on learning better Bayesian network classifiers. Among the many contributions contained in this monography, it is worth to highlight the following: A new classifier, called "Indifferent Naive Bayes" based on Bayesian model averaging and the principle of indifference, that improves the classical Naive Bayes classifier, and a series of three new algorithms for learning Tree Augmented Naive Bayes (TAN) models. These algorithms extend previous results by proving that, assuming a prior decomposable distribution over TANs, one can compute the exact Bayesian model averaging, over TAN structures and parameters, in polynomial time. It is also proved that the k-maximum a posteriori TAN structures can also be computed in polynomial time. These classifiers provide consistently better accuracies, over Irvine datasets as well as over artificially generated data, than the TAN-based classifiers previous ly reported in the literarture. Furthermore, from a practical point of view, these three TAN classifiers based on decomposable distributions can be seen as alternatives for different tradeoffs between accuracy and complexity.

Bayesian network learning is a hard area of research that requires mathematical sophistication and advanced programming skills. Dr. Jesús Cerquides, one of the most brilliant researchers I have ever met, largely fulfils both requirements. I have learnt a lot working with him and I have been very lucky having him as PhD student.

Bellaterra, October 2004

Ramon López de Mántaras
Institut d'Investigació en IntelÂ·ligència Artificial
Consell Superior d'Investigacions Científiques

Acknowledgements

It's a sign of mediocrity when you demonstrate gratitude with moderation.

Roberto Benigni

A few days before my defense is a good point in time to take a look back to the long road that led here and the friends that shared the way. Writing a doctoral thesis has required a big effort from me, but an even bigger amount of help from others.

First of all, I would like to thank my advisor, Ramon López de Màntaras. It was in my fourth year as undergraduate student, when I still did not know him personally, that I found and read a book authored by him in the faculty library. I thought it would be amazing to work with somebody like him. I did not know by that time that only two years later we would be coauthoring our first paper. He has taught me how to read and write a scientific paper. He has supported me in many ways and always has been there to listen to this and that new idea. Most importantly, he has always shown confidence in me and my work.

In 1997, I stayed for a semester at ISI, USC. I am thankful to Yolanda Gil for her support and kindness, to Gal Kaminka and José Luis Ambite for the interesting conversations about everything surrounding science and politics. I am specially in debt with Wei Min Shen, who showed me that simple is good. Simple questions can be the most relevant. Simple solutions can work better than complex ones. He pointed me to the work of E.T. Jaynes, and doing so he shed light in my dark way.

In Ubilab, UBS I learnt a lot about object orientation and programming from Dirk Riehle and Hans Wegener. Maria Luisa Barja was extremely supportive. I cannot forget the many coffees we shared complaining about who cares what.

Carlos Domingo introduced me to learning theory and taught me that there is nothing more practical than a good theory.

This thesis is presented while I am staying at the Department of Matemàtica Aplicada i Anàlisi in the University of Barcelona. I am thankful to Carles Simó for allowing me the use of the department computer facilities and not complaining after I was using about ten times what I said I would use. I am

also thankful to Joaquim Font, Jaume Timoneda and Anna Puig for their help and encouragement and for listening to me while complaining during the last months.

I am also grateful to Maite López-Sánchez, Jesús Vega and Juan Antonio Rodríguez-Aguilar for listening and helping all the way through.

I am and will always be in debt with my parents, who taught me how to live and act, by examples. And with my brother, who taught me about the noble art of discussion. They have always provided encouragement. They have always been a lighthouse to turn to when you lose your way.

Finally, all along this time you have been with me, Elena, thanks for standing there through the good times and the bad. Thanks for bringing me up so many times. And thanks for accepting sharing our room with a computer running learning algorithms all night long while we were in Switzerland.

A bit of this thesis comes from each one of you, feel free to choose what.

THANKS.

Abstract

This thesis applies objective Bayesian probability theory techniques to improve Bayesian network classifiers. The main contributions are:

- A parallelizable distance based discretization method that allows to extend discrete classifiers to non-discrete domains.
- The concepts of qualitative influences and synergies, which allow to improve understandability of Bayesian network classifiers.
- The first order and second order qualitative Bayesian classifiers, which are classifiers based on qualitative influences and synergies, with easily understandable results and with a reasonable accuracy.
- INDIFFERENTNB, an improved version of the naive Bayes classifier based on the naive Bayes model, naive distributions, Bayesian model averaging and the principle of indifference that improves the quality of the probabilities of the maximum likelihood naive Bayes classifier.
- STAN+BMA, a classification algorithm based on applying empirical local Bayesian model averaging to the STAN (softened TAN) classifier and which improves its accuracy.
- TBMATAN, a classification algorithm based on the computability of the averaging of TAN models under decomposable distributions over TANs and the principle of indifference that improves STAN accuracy.
- SSTBMATAN, an efficient approximation to TBMATAN which provides also improved accuracy.
- MAPTAN, a classification algorihtm that computes the maximum a posteriori TAN model and improves STAN accuracy.
- MAPTAN+BMA, a classification algorithm that computes the k maximum a posteriori TAN models and their relative weights efficiently and improves STAN accuracy and STAN+BMA accuracy and learning time.

These results show that the joint application of Bayesian model averaging and a careful selection of the prior probability distribution over the set of models, following objective Bayesian techniques whenever it is possible, can provide significant improvements to classification algorithms.

Chapter 1

Introduction

In every phenomenon the beginning remains always the most notable moment.

Thomas Carlyle

In colloquial terms, learning to classify consists in analyzing a set of objects with different characteristics and of different classes and after that being able to assign a possibly unseen object to one of the classes. The field of Machine Learning has been interested in designing automatic classification algorithms since its inception. The areas of application of such algorithms are immense, ranging from historic Artificial Intelligence (AI in the following) objectives such as "begin able to adapt the behaviour of machines by telling them what is good and what is bad" to far more recent and business oriented objectives such as "targeting direct marketing campaigns".

Probability theory has always been perceived as a relevant discipline to help in the quest of solving the classification problem and for AI in general. Furthermore, the development in the late eighties and early nineties in the field of Bayesian networks, together with an increased interest from the community, have further increased this belief. In this thesis we show several ways in which the application of probability theory can improve Bayesian network classifiers.

We start this chapter by introducing the objectives of the thesis in section 1.1. After that, we provide a roadmap for the reader in section 1.2 pointing out the key contributions of the thesis. Finally, in section 1.3 we outline a rationale for the contributions.

1.1 Objective

This thesis focuses in improving a family of classification algorithms known as *Bayesian network classifiers*. The objective of the thesis is improving Bayesian network classifiers by means of the application of objective Bayesian probability theory techniques.

1.2 Roadmap and contributions

In this section we provide an overview of the structure of the thesis which is depicted in Figure 1.1. In that figure, chapters with original contributions are dark gray coloured while introductory and review chapters appear coloured light gray.

The thesis starts with this introductory chapter, where the reader can find the objectives, structure and a rationale for the contributions.

Chapter 2 presents a short introduction to the foundations of probability theory summarizing the results in the first two chapters of "Probability Theory: The logic of science" by E.T. Jaynes. The reason for including this short summary is twofold: on one hand it will help a reader without knowledge about Bayesian statistics understand the philosophy behind the thesis, on the other hand, it gives a deserved visibility to these results. Also in chapter 2 the principle of indifference is introduced. The principle will be used in the contributions of chapters 6 and 8.

In order to ease understanding of the contributions of the thesis, chapter 3 contains an introduction to the problem of classification, a short state of the art of Bayesian network classifiers and a presentation of Bayesian model averaging as a probability technique that is useful for the design of classifiers. The notation and terminology to be used in the thesis are fixed in this chapter. Also in this chapter, the two main classification algorithms that have been improved are introduced: Naive Bayes and Tree Augmented Naive Bayes (TAN).

Discretization techniques help broaden the application of classifiers that cannot deal easily with numerical attributes. In chapter 4 we present a discretization method that can be implemented in parallel providing for a fast and effective method for the discretization of numerical attributes and prove its performance against state of the art discretization methods. The work has been presented at the Third International Conference on Knowledge Discovery and Data Mining.

Chapters 5 and 6 will present two different ways of improving the Naive Bayes classifier. In chapter 5 a variation of the naive Bayes classifier is introduced which is easier to interpret while keeping a reasonable accuracy. This work has been presented at the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery. In chapter 6 Bayesian model averaging and the principle of indifference are applied in order to construct a more accurate Naive Bayes classifier. This work has been presented at The 16th International FLAIRS Conference.

Chapters 7, 8 and 9 show different ways in which Bayesian techniques can

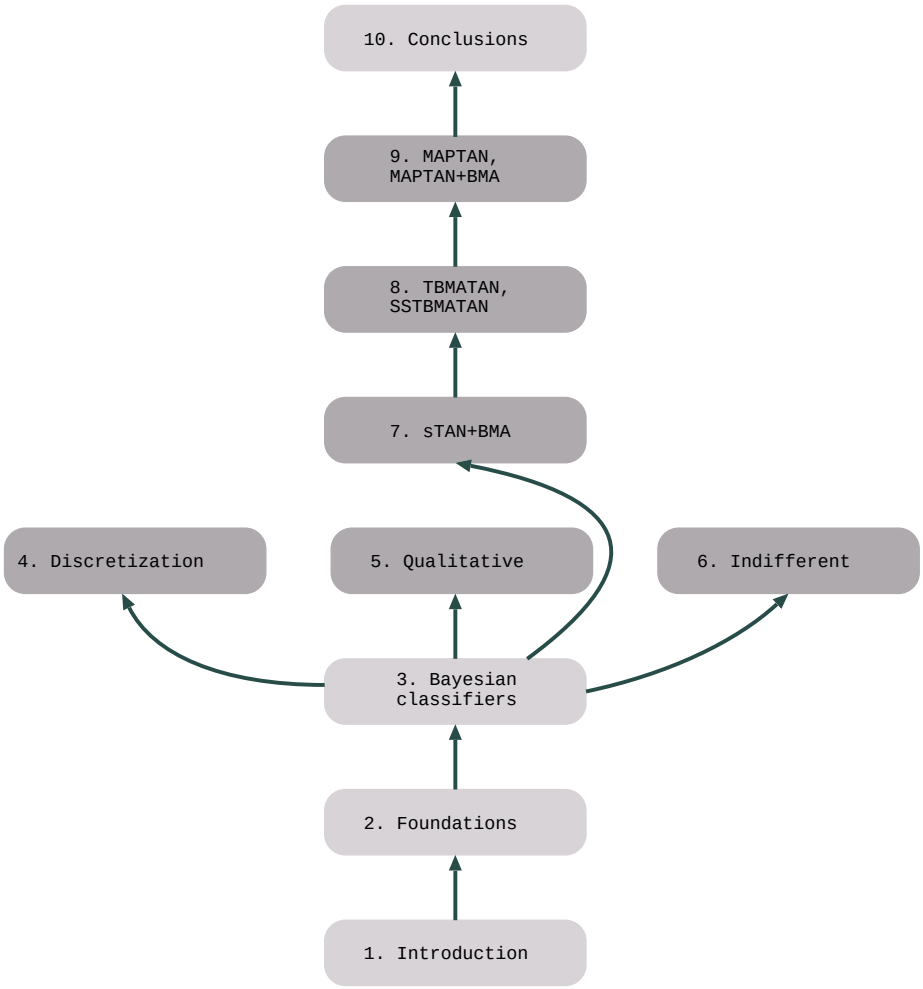


Figure 1.1: Thesis roadmap

improve Tree Augmented Naive Bayes. Chapter 7 shows a hands-on approach to the calculation of the model averaging of TAN by the use of empirical local Bayesian model averaging that results in a classifier (STAN+BMA) that improves both the classification accuracy and the approximation of the class probabilities of the state of the art TAN classifier (Friedman et al., 1997): STAN. This work has been presented at the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Chapter 8 presents the most significant result in this thesis, showing that under some conditions, the averaging of models for Tree Augmented Naive Bayes results in an integral that can be calculated in closed form. To do this we introduce decomposable distributions over TANs and show that the expression resulting from the Bayesian model averaging of TAN models can be integrated into closed form if we assume the prior probability distribution to be a decomposable distribution. This result allows the construction of a classifier (TBMATAN) that is most of the cases more accurate than STAN and approximates better the class probabilities. TBMATAN learning time can be very long for large datasets due to computational problems. To fix this problem we introduce an approximation to TBMATAN (SSTBMATAN) which is more efficient. SSTBMATAN has a shorter learning time a longer classification time than STAN. SSTBMATAN is most of the cases more accurate than TBMATAN and approximates better the class probabilities. This work has been presented at the Twentieth International Conference on Machine Learning. Finally, in chapter 9, we show that it is possible to calculate efficiently both the TAN model with maximum a posteriori probability, and the set of k TAN models with maximum a posteriori probability and their relative probability weights. We show that these results allow the construction of two classifiers (MAPTAN and MAPTAN+BMA) which outperform STAN and STAN+BMA respectively in error rate and quality of the predicted probabilities. Furthermore, MAPTAN+BMA learning time complexity is lower than STAN+BMA. In the three chapters, experimental results are given that allow the reader to evaluate the improvements.

1.3 Rationale for the contributions

In this section we propose a simplified comparison of classifiers from the perspective of a user and motivate the contributions of this thesis in this simplified framework. The purpose of the section is easing understanding of what the thesis proposes, why we think it can be useful and what have been the main characteristics we have taken into consideration while trying to improve Bayesian classifiers. The reader should not take this framework as a proposal for future use but as a simple tool for understanding what has been done.

1.3.1 Characteristics of a classifier

Classifiers can be compared by different characteristics. In order to ease the understanding of the contributions of the thesis from the point of view of a user of machine learning or data mining algorithms, at the end of some chapters the

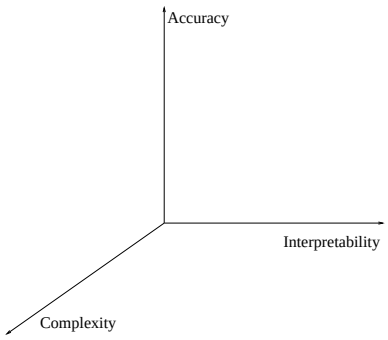


Figure 1.2: Three axis in which a classifier can be placed

different classifiers introduced are plot in a three dimensional space where the three axis are determined by accuracy, interpretability and time complexity (see Figure 1.2). Evidently, these are not the only characteristics we could define, but in our opinion these are the most significant when evaluating a classifier. For a more extended list of classifier characteristics from the perspective of a user, the reader can refer to section 10.7 of (Hastie et al., 2001).

Accuracy can be defined as how well a classifier performs the task that is its main objective, that is, classifying. There are different measures of classification accuracy. The error rate computes the percentage of instances that are misclassified. Some more sophisticated measures do also take into account the probability that the classifier has assigned to the class. This way, those measures punish much more misclassifications where the probability of the class was very high than those were the classifier itself knew that it was confused and assigned a not so high probability to the most probable class. In the last few years, the evaluation of accuracy of classifiers based on the Receiver Operating Characteristics is gaining momentum (Provost et al., 1998; Fawcett, 2003). Independently of how it is measured, increasing classification accuracy is, obviously, an objective when proposing new classifiers.

Interpretability stands for how easily humans can understand the decisions taken by a classifier. Many times, machine learning algorithms, and also classifiers, are used as supporting devices for a human being who is in charge of taking a decision in a concrete situation. In these cases, being able to interpret, understand and explain to interested third parties the rationale behind the decision will influence heavily whether we can use a classifier or not. Increasing the interpretability of classifiers is hence another objective when proposing new classifiers.

Time complexity is the amount of time needed for a classifier to perform its task. Given a dataset, classifiers usually split their work in two steps, a learning step where the dataset is processed and the information from the dataset summarized in some way, and a classification step where new unclassified instances are observed and the classifier guesses the class in which they should be classi-

fied. This split generates two different complexity measures for a classifier. We will call learning time complexity the time spent on the first task, and classification time complexity the time spent to classify a new instance once the learning step is over. Decreasing both learning time complexity and classification time complexity is yet another objective when proposing new classifiers.

1.3.2 Why do we need new classifiers?

If we accept the simplified view proposed by this three axis, when proposing a new classifier we can be in one of two cases. In the first case, we can move in a positive direction in one or more of the axis while keeping the others fixed. This is the case, for example, when the accuracy of a classifier is increased without modifying its interpretability or time complexity, which is the case for the improvement of Naive Bayes presented provided by INDIFFERENTNB (see chapter 6) as well as for the improvement for STAN provided by MAPTAN (see chapter 9). This kind of "moves" are clear improvements of previous results and are usually expected to substitute the previous version of the classifier. In the second case, we can move in a positive direction in some axis while losing in some other. This is the case, for example, when we create a classifier with a better accuracy but at the cost of increasing the time complexity and decreasing the interpretability, as is the case for the improvements for STAN suggested in chapters 7 and 8. This is also the case when we create a classifier with increased interpretability but higher time complexity, as is the case for the improvement presented for Naive Bayes in chapter 5. This kind of moves provide an increase in the "palette" of tools for the classification user. In the face of a new problem, the user can evaluate which are the constraints in that concrete case for accuracy, interpretability and complexity and choose the classifier best suited for the task. For example, imagine that our task is to create a system to be used for disease self-treatment and that the system has to suggest the user the correct treatment for his disease. Assuming the user is not knowledgeable in the area, he will have to rely on the system decision without trying to interpret the decision, because he should be knowledgeable in medicine to do that. He will not be willing to wait for days to get an answer, but he can wait for one or two hours. Accuracy is probably the most important characteristic for a classifier for this task. If we slightly change the task and assume that the classifier will be used by a doctor in his office, this heavily increases the interpretability and time complexity needs (he will definitely not be willing to make the patient wait for the computer to answer back) and lowers the accuracy needs, because probably the doctor will use the system as a double check for his own decisions and in case of disagreeing he will be able to explore the patient more carefully and generate the adequate treatment.

Due to the increasing computational power, that allows higher complexity calculations, increasing accuracy is specially significant. That is why we consider specially valuable the results in chapter 8, which provide, to the best of our knowledge, the most accurate classifier with a learning time complexity linear on the number of observations of the dataset.

We understand that the new classifiers proposed in this thesis provide reasonable trade-offs and cover areas of the described three dimensional space that where uncovered before. Hence we think that they deserve to be selected by any user for some domains and that they could end up being a valuable tool in an users “palette”.

Chapter 2

Bayesian foundations of the learning process

> *. . . instead of asking, "How can we build a mathematical model of human common sense?" let us ask, "How could we build a machine which would carry out useful plausible reasoning, following clearly defined principles expressing an idealized common sense?"*
>
> E.T. Jaynes

Since Laplace wrote his famous sentence "Probability theory is nothing but common sense reduced to calculation" there has been a part of the research community that has regarded probability theory as an extension of logic. This chapter explains the main results of the work of E.T. Jaynes, R.T. Cox and others in trying to see probability theory in this role. Including them here we try to honour and foster these ideas, while providing a theoretical foundation for some of the developments in the thesis.

2.1 Probability theory as extended logic

Considering probability theory as extended logic means accepting that "there is only a single set of rules for doing plausible reasoning which is consistent and in qualitative correspondence with common sense" (Jaynes, 1988).

Nowadays, almost everybody accepts Aristotelian logic as "the" deductive logic. It has been with us for more than two thousand years and stands at the basis of all sciences. Will there ever be such agreement in the inductive logic arena? Jaynes' answer is that there will be, and that probability theory as presented in the following will be finally be accepted as "the" inductive logic.

In the following we will review how probability theory can be derived just by setting logical goals. Prior to this we introduce some notation and vocabulary. In the following, *degree of plausibility* means degree of belief, how much do we trust or belief in something. Let A, B, C represent propositions. $A|B$ represents the conditional plausibility that A is true given that B is true.

2.1.1 The basic desiderata

We will set three conditions over our degrees of plausibility.

1. Degrees of plausibility are represented by real numbers.

2. Our reasoning must correspond qualitatively with common sense.

3. Our reasoning should be consistent:

   (a) If a conclusion can be reached in more than one way, every possible way should lead to the same result.

   (b) Our reasoning must include deductive logic as a special case. In the limit where propositions become certain or impossible in any way, every equation must reduce to a valid example of deductive reasoning.

2.1.2 Justifying the desiderata

Our stated desiderata are intuitively compulsory for any inductive logic worth its name. We will try to justify it in the following while deepening into the understanding of how these desiderata translate into mathematical constraints for our logic. Following the notation in (Keynes, 1921), (Cox, 1961) and (Jaynes, 1996), the symbol

$$A|B$$

indicates the conditional plausibility that A is true given that B is true. Parentheses carry no semantic and will be used just to clarify the grouping of expressions.

Degrees of plausibility are represented by real numbers

Our first condition seems very intuitive. Anyhow, if it looks too restrictive, the same results can be obtained by replacing it with two more elementary ones:

1. We should have a way of comparing plausibilities such that:
   (a) Is transitive. If $(A|X) \geq (B|X)$ and $(B|X) \geq (C|X)$then $(A|X) \geq (C|X)$.
   (b) Is Universal. Given A, B, C then one of the relations $(A|C) > (B|C)$, $(A|C) = (B|C)$, $(A|C) < (B|C)$ must hold.

For a detailed discussion of this equivalence and the reasons for accepting universal comparability (1b) see Appendix A of (Jaynes, 1996).

Our reasoning must correspond qualitatively with common sense

Our second condition means that our theory must, on a qualitative scale, be consistent with common sense rules. In order to clarify how this common sense rules look like, we will present an example. Let A, B, C, D be propositions. If the plausibility of A is bigger given D than given C, that is

$$(A|D) > (A|C)$$

but the plausibility of B given A is the same in both cases

$$(B|AD) = (B|AC)$$

then common sense tells us that we cannot have a decrease in the plausibility that both A and B are true

$$(AB|D) \geq (AB|C)$$

and there should be a decrease in the plausibility that A is false

$$(\overline{A}|D) < (\overline{A}|C)$$

These qualitative requirements only set the directions in which the plausibilities should go, not how much. It is also accepted as condition that an infinitesimal difference between $(A|D)$ and $(A|C)$ will only induce an infinitesimal difference between $(AB|D)$ and $(AB|C)$ and between $(\overline{A}|D)$ and $(\overline{A}|C)$.

Our reasoning should be consistent

Our third condition probably needs less justification than the previous, since consistence has been a common condition for most logics. As previously explained, Aristotelian logic is accepted as "the" deductive logic. We want our logic to be consistent with it. In a setting where Aristotelian logic can be used, the usage of probability theory as logic should give the same results.

2.2 The quantitative rules

We have established the conditions which from our point of view cannot be waived for an inductive logic. In the following we will see how these conditions uniquely determine the rules of probability theory.

2.2.1 The product rule

Here we will look for a way of obtaining the plausibility of AB from the plausibility of A and B separately. What we are looking for is a function F such that

$$(AB|C) = F[Pl_1, Pl_2] \tag{2.1}$$

where Pl_1 and Pl_2 are different possible ways of separately calculating the plausibility of A and B. First of all we will try to identify what are the options for Pl_A and Pl_B. In order to do this we will try all the different alternatives and discard those that exhibit violations of common sense. We define

$$u \equiv (AB|C), \quad v \equiv (A|C), \quad w \equiv (B|AC), \quad x \equiv (B|C), \quad y \equiv (A|BC)$$

And we would like u to be expressed as a function of two or more of v, w, x, y. In (Tribus, 1969) it is shown that all but two possibilities can exhibit violations of common sense in some extreme case. Those two are $u = F(x, y)$ and $u = F(v, w)$. Hence, we can conclude that the two unique ways of decomposing $(AB|C)$ are

$$\begin{array}{l}(AB|C) = F[(B|C), (A|BC)] \\ (AB|C) = F[(A|C), (B|AC)]\end{array} \tag{2.2}$$

These two are in fact equivalent but for the naming of the propositions. We can apply this decomposition to $(ABC|D)$ in two different ways:

$$(ABC|D) = F[(BC|D), (A|BCD)] = F\{F[(C|D), (B|CD)], (A|BCD)\} \tag{2.3}$$

but also

$$(ABC|D) = F[(C|D), (AB|CD)] = F\{(C|D), F[(B|CD), (A|BCD)]\} \tag{2.4}$$

According to desideratum (3a), both ways of reasoning must give the same result. This means that F must fulfill the functional equation [1]:

$$F[F(x, y), z] = F[x, F(y, z)] \tag{2.5}$$

[1] A technical detail needs to be fixed before this sentence is true. It has been reported by Halpern (Halpern, 1999a; Halpern, 1999b) that there is a finite model not satisfying this step. The problem in the demonstration has been fixed by Arnborg and Sjödin (Arnborg and Sjödin, 1999; Arnborg and Sjödin, 2000b; Arnborg and Sjödin, 2000a) by noticing that we should impose "extensibility" in order for this step to be true for non-dense (for example finite) models

Now we apply the qualitative correspondence with common sense requirement explained in the section 2.1.2. This imposes that

$$\begin{gathered} F(x,y) \text{ must be a continuous monotonic strictly} \\ \text{increasing function of both } x \text{ and } y \end{gathered} \tag{2.6}$$

In this setting, the general solution to equation 2.5 can be expressed in two forms:

$$\begin{gathered} w[F(x,y)] = w(x)w(y) \\ F(x,y) = w^{-1}[w(x)w(y)] \end{gathered} \tag{2.7}$$

where $w(x)$ is an arbitrary function. A long proof of this result can be found in (Aczel, 1966). A shorter one, assuming F is differentiable can be found in (Cox, 1961; Jaynes, 1996).

Therefore our combination rule has two possible forms:

$$w(AB|C) = w(A|C)w(B|AC) \tag{2.8}$$

$$w(AB|C) = w(B|C)w(A|BC) \tag{2.9}$$

Due to equation 2.6 $w(x)$ must be a continuous monotonic function. Condition (3b) places further restrictions on $w(x)$. Assume that A is certain given C. Then, assuming C is true, the propositions AB and B are the same (one is true if and only if the other is true). This means that $(AB|C) = (B|C)$. Reasoning similarly we have that $(A|BC) = (A|C)$. In this case, equation 2.8 reduces to:

$$w(B|C) = w(A|C)w(B|C) \tag{2.10}$$

which implies that

$$\text{Certainty must be represented by } w(A|C) = 1 \tag{2.11}$$

Now assume A is impossible, given C. Then $(AB|C) = (A|C)$ and if B does not contradict C, then $(A|BC) = (A|C)$. In this case equation 2.8 reduces to:

$$w(A|C) = w(A|C)w(B|C) \tag{2.12}$$

that should hold no matter what is the plausibility of B. This means that

$$\text{Impossibility should be represented either by } w(A|C) = 0 \text{ or by } w(A|C) = +\infty \tag{2.13}$$

These two possibilities are in fact isomorphic just by using the transformation $w'(x) = \frac{1}{w(x)}$. Therefore we can choose to represent impossibility by 0 as a convention. What we still do not know is how $w(x)$ varies between 0 and 1.

2.2.2 The sum rule

Here we will look for a way of relating the plausibility of A with the plausibility of $\overline{A}$. Defining $u \equiv w(A|B)$ and $v \equiv w(\overline{A}|B)$ we are looking for a functional relation

$$v = S(u) \tag{2.14}$$

Qualitative correspondence with common sense requires $S(u)$ to be a continuous monotonic decreasing function in $0 \leq u \leq 1$, with extreme values $S(0) = 1$, $S(1) = 0$. S must also be consistent with

$$w(AB|C) = w(A|C)w(B|AC) \tag{2.15}$$

$$w(A\overline{B}|C) = w(A|C)w(\overline{B}|AC) \tag{2.16}$$

Combining equations 2.14,2.15,2.16 we get

$$w(AB|C) = w(A|C)S\left[\frac{w(A\overline{B}|C)}{w(A|C)}\right] \tag{2.17}$$

and using commutativity with respect to A and B we get

$$w(A|C)S\left[\frac{w(A\overline{B}|C)}{w(A|C)}\right] = w(B|C)S\left[\frac{w(B\overline{A}|C)}{w(B|C)}\right] \tag{2.18}$$

Particularly, if we take $\overline{B} = AD$ where D is any new proposition we have

$$w(A\overline{B}|C) = w(\overline{B}|C) = S[w(B|C)] \tag{2.19}$$

$$w(B\overline{A}|C) = w(\overline{A}|C) = S[w(A|C)] \tag{2.20}$$

Using, $x \equiv w(A|C)$ and $y \equiv w(B|C)$ equation 2.18 becomes

$$x\ S\left[\tfrac{S(y)}{x}\right] = y\ S\left[\tfrac{S(x)}{y}\right] \quad , \quad \begin{matrix} 0 \leq S(y) \leq x \\ 0 \leq x \leq 1 \end{matrix} \tag{2.21}$$

The most general function satisfying equation 2.21 with the boundary condition $S(0) = 1$ is already available in the literature:

$$S(x) = (1 - x^m)^{\frac{1}{m}} \tag{2.22}$$

Once again, as happened with equation 2.7 a long demonstration of this result can be found in (Aczel, 1966) and a shorter one, assuming S is twice differentiable, can be found in (Cox, 1961; Jaynes, 1996). It can be shown that equation 2.22 is the necessary and sufficient condition on $S(x)$ for consistency in the sense of equation 2.18

Summarizing, what we have found so far is that there should exist a continuous monotone function $w(x)$ fulfilling

$$w(AB|C) = w(A|C)w(B|AC) = w(B|C)w(A|BC) \tag{2.23}$$

$$w^m(A|B) + w^m(\overline{A}|B) = 1 \tag{2.24}$$

But we can rewrite 2.23 as

$$w^m(AB|C) = w^m(A|C)w^m(B|AC) = w^m(B|C)w^m(A|BC) \tag{2.25}$$

then we notice that the value of m is irrelevant. We can always define $p(x) = w^m(x)$ and be sure that

For every reasoning process that is consistent and maintains qualitative correspondence with common sense, there exists a continuous monotone function $p(x)$ such that:

$$\begin{gathered} \forall x \quad 0 \leq p(x) \leq 1 \\ p(A|B) + p(\overline{A}|B) = 1 \\ p(AB|C) = p(A|C)p(B|AC) = p(B|C)p(A|BC) \end{gathered} \tag{2.26}$$

Rules in equation 2.26 offer an infinite number of ways for doing plausible reasoning, corresponding to every different choice of $p(x)$. In the next section we will show that under certain conditions, $p(x)$ can be uniquely determined.

2.3 Prior probabilities

The rules appearing in equation 2.26 provide the basis for plausible reasoning. In order to reason with these rules, we need to have information that allows us to determine the values of $p(x)$ for a subset of the propositions and then use the rules to calculate the values of $p(x)$ for the proposition we are interested in. The problem of determining prior probabilities is key to the theory of probability as extended logic, as has been noticed in (Jaynes, 1968). A good review of the field is done in (Kass and Wasserman, 1994). In this section we will shortly review some of the rules for determining prior distributions.

2.3.1 The principle of indifference

The principle of indifference give us the way of assigning values to probabilities in the simplest case, when we are given a set of alternatives and have no reason to prefer anyone of them.

Suppose we have a set of propositions $\{A_1, \ldots, A_n\}$ that are mutually exclusive (not two of them can be simultaneously true) and exhaustive (one of them must be necessarily true) given B. In this situation, we can deduct from the rules appearing in 2.26 that

$$\sum_{i=0}^{n} p(A_i|B) = 1 \tag{2.27}$$

But this is still not enough to assign concrete values to the probabilities. Suppose that information B is indifferent between every two pairs of propositions A_i,A_j, that is, whatever it says about A_i, it says the same thing about A_j, and so it contains nothing that gives any reason to prefer one over the other. Under these conditions, and using the consistency desideratum we have that

$$p(A_i|B) = \tfrac{1}{n} \qquad 1 \leq i \leq n \tag{2.28}$$

as appears in (Jaynes, 1996; Jaynes, 1988). This is the simplest case where we can get a set of definite numerical values for $p(x)$ that is uniquely determined

by the desiderata that appear in section 2.1.1. As a *user guide* for the principle of indifference, we will give the conditions under which we can use it:

1. We must be able to analyze the situation into an enumeration of the different possibilities which we recognize as mutually exclusive and exhaustive.

2. Having done this, we must then find that the available information gives us no reason to prefer any possibility to any other.

When these two conditions are met we can use the result in equation 2.28. It is important to note that condition 2 can be satisfied in two different ways: it may be the consequence of positive knowledge or the consequence of complete ignorance. It is often the case that condition 2 is not satisfied. Then we have to use other ways of determining priors, such as the entropy principle.

2.3.2 The entropy principle

Suppose we have a six faced die for which we have the information that the average number of spots over a set of trials is 4 instead of 3.5 as is expected from a fair one. Which are the probabilities of each face? In this case we cannot apply the principle of indifference and assign an equal probability to each face, because this will contradict the information we have available, providing an average of 3.5. The entropy principle is based on the idea that, in cases like this, we should select the prior distribution that takes into account the information we have available "and nothing else".

In order to do this, we have to have a measure of the amount of uncertainty in a probability distribution. Let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be again a exhaustive set of mutually exclusive propositions given B. It can be shown (Jaynes, 1996), using a logical way of reasoning similar to the ones in the previous sections, that the expression

$$H(\mathcal{A}|B) = -\sum_{i=1}^{n} p(A_i|B)\log(p(A_i|B)) \tag{2.29}$$

is the only *reasonable* measure of "amount of uncertainty". Thus, what the entropy principle states is that

For given information that places *any* definite kind of constraint on the problem, we should select as a priori probability distribution the one which maximizes 2.29 while fulfilling the constraints

In practice, this is usually solved by an application of nonlinear programming methods such as Lagrange multipliers.

2.3.3 Other methods

Both the indifference and the entropy principles are applicable to discrete probability spaces. In the literature we can find methods for continuous probability

spaces, but the reasons behind them are from our point of view not so logically compelling as the ones for the methods previously seen.

Between those methods we would like to highlight priors based on the application of group invariance arguments. In this family of priors we need to define transformations for which we want our results to be invariant. One of the simplest cases, is the estimation of the parameters of a normal distribution $N(\mu, \sigma)$. If we are estimating a location parameter, we should be invariant to scaling and translation. This means that, if given the set of observations $\{x_1, \ldots, x_n\}$ we estimate $\mu = \mu_1$ and $\sigma = \sigma_1$ then given $\{Ax_1 + B, \ldots, Ax_n + B\}$ we should estimate $\mu = \mu_1 + B$ and $\sigma = A\sigma_1$. Formalizing this correctly in terms of group invariance give us that the prior over (μ, σ) should be $p(\mu, \sigma) = \frac{1}{\sigma^2}$. More information about this family of priors can be found in (Dawid, 1983; Hartigan, 1964; Jaynes, 1968).

Especially significant is also the work of Bernardo in Bayesian reference analysis. Quoting Bernardo's (Bernardo and Ramón, 1998):

> The declared objective of reference Bayesian analysis is to specify a prior distribution such that, even for moderate sample sizes, the information provided by the data should dominate the prior information because of the vague nature of the prior knowledge. Reference analysis uses the concept of statistical information (...) The amount of information to be expected from an experiment about some quantity of interest naturally depends on the available prior knowledge: the more prior information available, the less information may be expected to be learned from the data. An infinitely large experiment would eventually provide all missing information; thus, it is possible to obtain a measure of the amount of missing information as a limiting form of a functional of the prior distribution. It is natural to define vague prior knowledge as that with the largest missing information: the reference prior should then be that which maximizes the missing information.

Bernardo's technique for determining priors is so general that the principle of indifference and the entropy principle can be seen as applications of reference analysis in their concrete cases. A nicely written introduction to the technique can be found in (Bernardo, 1998).

2.4 Summary

In this chapter we have reviewed the development of the rules of probability from logical constraints. Afterwards we have reviewed the problem of prior probabilities and rules for assigning priors in some concrete cases. The main idea behind this chapter is to explain and support Laplace's view of probability theory as the symbolic logic of plausible reasoning. In this sense, for any learning algorithm, either probability theory should lay at its foundation, or the reasoning that is performed in it can be inconsistent in some cases. The idea is by no means

new and is gaining widespread acceptance in different disciplines, as can be seen in (Domingos, 1997; Hanson et al., 1991; Hoeting et al., 1998; Jaynes, 1996; MacKay, 1995; Shen, 1993).

Chapter 3

Bayesian network classifiers

A tree growing out of the ground is as wonderful today as it ever was. It does not need to adopt new and startling methods.

Robert Henri

Bayesian network classifiers are classification algorithms that assume probability distributions encoded as Bayesian networks as the probabilistic model for the dataset that is the learning objective. This chapter first introduces the notation to be used in the thesis, which is an attempt to put together the different notations used in (Cerquides, 1999a), (Heckerman et al., 1995), (Friedman et al., 1997) and (Meila and Jaakkola, 2000a) and some conventions in the machine learning literature. After that it presents the two Bayesian network classifiers which have attracted most of the attention in the literature: naive Bayes and Tree Augmented Naive Bayes (TAN from now on). Finally, the chapter introduces Bayesian model averaging, a probabilistic technique that will be applied in Chapters 6, 7 and 8 to improve both naive Bayes and TAN.

3.1 The problem of classification

A *discrete attribute* is a finite set, for example we can define attribute $Pressure$ as $Pressure = \{Low, Medium, High\}$. A *discrete domain* is a finite set of discrete attributes. We will note $\Omega = \{X_1, \ldots, X_m\}$ for a discrete domain, where $X_1, \ldots, X_m$ are the attributes in the domain. A *classified discrete domain* is a discrete domain where one of the attributes is distinguished as "class". We will use $\Omega_C = \{A_1, \ldots, A_n, C\}$ for a classified discrete domain. In the rest of the thesis we will refer to an attribute either as X_i (when it is considered part of a discrete domain), A_i (when it is considered part of a classified discrete domain and it is not the class) and C (when it is the class of a classified discrete domain). We will note as $V = \{A_1, \ldots, A_n\}$ the set of attributes in a classified discrete domain that are not the class.

Given an attribute A, we will note $\#A$ as the number of different values of A. We define $\#\Omega = \prod_{i=1}^{m} \#X_i$ and $\#\Omega_C = \#C \prod_{i=1}^{n} \#A_i$.

An *observation* x in a classified discrete domain Ω_C is an ordered tuple $x = (x_1, \ldots, x_n, x_C) \in A_1 \times \ldots \times A_n \times C$. An *unclassified observation* S in Ω_C is an ordered tuple $S = (s_1, \ldots, s_n) \in A_1 \times \ldots \times A_n$. To be homogeneous we will abuse this notation a bit noting s_C for a possible value of the class for S. A *dataset* $\mathcal{D}$ in Ω_C is a multiset of classified observations in Ω_C.

We will note N for the number of observations in the dataset. We will also note $N_i(x_i)$ for the number of observations in $\mathcal{D}$ where the value for A_i is x_i, $N_{i,j}(x_i, x_j)$ the number of observations in $\mathcal{D}$ where the value for A_i is x_i and the value for A_j is x_j and similarly for $N_{i,j,k}(x_i, x_j, x_k)$ and so on. We note similarly $f_i(x_i), f_{i,j}(x_i, x_j), \ldots$ the frequencies in $\mathcal{D}$. It is worth noticing that f defines a probability distribution over $A_1 \times \ldots \times A_n \times C$.

A *classifier* in a classified discrete domain Ω_C is a procedure that given a dataset $\mathcal{D}$ in Ω_C and an unclassified observation S in Ω_C assigns a class to S.

A *model* is a mathematical representation of reality that allows us to draw conclusions (that do not necessarily have to hold back in reality). A *probabilistic model* is a mathematical model that uses probability theory and hence allows us to draw probabilistic conclusions. We will note a model as M. Usually a model belongs to a family or set of models which we will note $\mathcal{M}$. Our setting for the learning problem will usually assume the learning algorithm is given a dataset $\mathcal{D}$ which is an independent identically distributed sample from an unknown probabilistic model M belonging to a known family of models $\mathcal{M}$. From this information, the learning algorithm should be able to classify unseen observations generated randomly from the model M.

3.2 Bayesian networks for classification

Bayesian networks are a valuable tool for solving the discrete classification problem. The approach that can be followed to apply them is to define a random variable for each attribute in Ω (the class is included but not distinguished at

this time). We will note $\mathbf{U} = \{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ where each $\mathcal{X}_i$ is a random variable over its corresponding attribute X_i. We extend the meaning of this notation to $\mathcal{A}_i$, $\mathcal{C}$ and $\mathcal{V}$. A *Bayesian network* over $\mathbf{U}$ is a pair $B = \langle G, \Theta \rangle$. The first component, G, is a directed acyclic graph whose vertices correspond to the random variables $\mathcal{X}_1, \ldots, \mathcal{X}_m$ and whose edges represent direct dependencies between the variables. The graph G encodes independence assumptions: each variable $\mathcal{X}_i$ is independent of its non-descendants given its parents in G. The second component of the pair, namely Θ, represents the set of parameters that quantifies the network. It contains a parameter $\theta_{i|\Pi_i}(x_i, \Pi_{x_i}) = P_B(x_i|\Pi_{x_i})$ for each $x_i \in X_i$ and $\Pi_{x_i} \in \Pi_{X_i}$, where Π_{X_i} denotes the Cartesian product of every X_j such that $\mathcal{X}_j$ is a parent of $\mathcal{X}_i$ in G. Π_i is the list of parents of $\mathcal{X}_i$ in G. We will note $\overline{\Pi_i} = \mathbf{U} - \{\mathcal{X}_i\} - \Pi_i$. A Bayesian network defines a unique joint probability distribution over $\mathbf{U}$ given by

$$P_B(x_1, \ldots, x_m) = \prod_{i=1}^{m} P_B(x_i|\Pi_{x_i}) = \prod_{i=1}^{m} \theta_{i|\Pi_i}(x_i|\Pi_{x_i}) \tag{3.1}$$

The application of Bayesian networks for classification can be very simple. For example suppose we have an algorithm that given a classified discrete domain Ω_C and a dataset $\mathcal{D}$ over Ω_C returns a Bayesian network B over $\mathbf{U} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{C}\}$ where each $\mathcal{A}_i$ (respectively $\mathcal{C}$) is a random variable over A_i (respectively C). Then if we are given a new unclassified observation S we can easily classify S into class $\underset{s_C \in C}{argmax}(P_B(s_1, \ldots, s_n, s_C))$. This simple mechanism allows us to see any Bayesian network learning algorithm as a classifier.

3.2.1 Dirichlet distributions

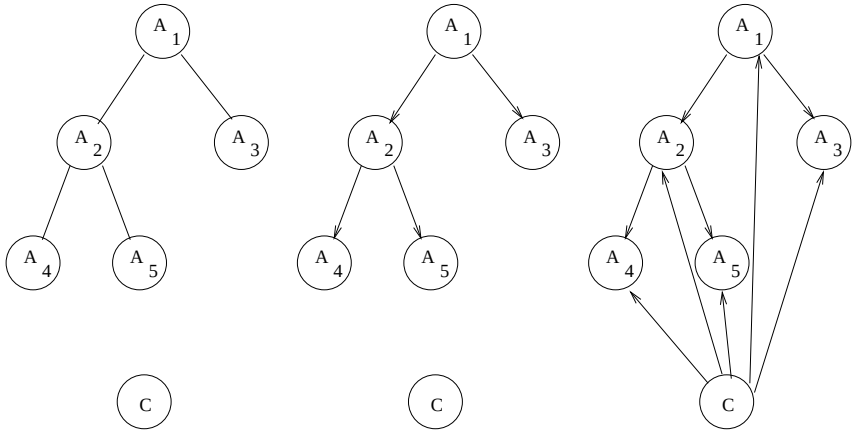
The Dirichlet probability distribution is frequently used in Bayesian networks because it is closed under multinomial sampling (Heckerman et al., 1995). In this thesis we will use the Dirichlet distribution as the basis for decomposable distributions over TANs, to be presented in chapter 8. It is defined over the parameter space $\theta_1, \ldots, \theta_k$, $(\sum_{i=1}^{k} \theta_i = 1,\ \theta_i > 0\ ,\ i = 1, \ldots, k)$ as:

$$D(\theta_1, \ldots, \theta_k; N_1, \ldots, N_k) = \frac{1}{Z_D} \prod_{i=1}^{k} {\theta_i}^{N_i - 1} \tag{3.2}$$

The numbers $N_1, \ldots, N_k$ are the hyperparameters of the Dirichlet distribution. The normalization constant Z_D has the form:

$$Z_D = \frac{\prod_{i=1}^{k} \Gamma(N_i)}{\Gamma(\sum_{i=1}^{k} N_i)} \tag{3.3}$$

with Γ denoting the Euler function $\Gamma(p) = \int_0^\infty x^{p-1} e^{-x} dx$



(a) E (b) $\overline{E}$ (c) $\overline{E}^*$

Figure 3.1: Notation for learning with trees

3.3 Naive Bayes

The naive Bayes classifier (Langley et al., 1992) is a classification method based on the assumption of conditional independence between the different variables in the dataset given the class.

If we assume that our data is generated by a naive Bayes model M, the operation of the model can be explained as follows. Given an unclassified observation S we must choose the class s_C that maximizes $P(\mathcal{C} = s_C|\mathcal{V} = S, M)$. Developing this conditional probability according to the Bayes rule we have:

$$P(\mathcal{C} = s_C|\mathcal{V} = S, M) = P(\mathcal{C} = s_C|M) * \frac{P(\mathcal{V} = S|\mathcal{C} = s_C, M)}{P(\mathcal{V} = S|M)} \tag{3.4}$$

Since M includes the assumption that the attributes are statistically independent given the class, it holds that:

$$P(\mathcal{V} = S|\mathcal{C} = s_C, M) = \prod_{i=1}^{n} P(\mathcal{A}_i = S_i|\mathcal{C} = s_C, M) \tag{3.5}$$

where n is the number of attributes of the dataset. Since $P(\mathcal{V} = S|M)$ does not depend on the class, naive Bayes tells us to choose the class s_C which maximizes:

$$P(\mathcal{C} = s_C|M) * \prod_{i=1}^{n} P(\mathcal{A}_i = S_i|\mathcal{C} = s_C, M) \tag{3.6}$$

The former equations describe in a very general sense the naive Bayes model and can be concreted into several different classification algorithms depending

on additional assumptions. In chapter 6 we will review some of these algorithms and present a well founded development for learning under the assumption that the model generating the data is a naive Bayes model.

3.4 Learning with trees

Given a classified domain Ω_C we will note $\mathcal{E}$ the set that contains every undirected graph E having $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ as nodes such that E is a tree. We will note as $\overline{E}$ a directed tree for E and $\overline{\mathcal{E}} = \{\overline{E} | E \in \mathcal{E}\}$. We will use $u, v \in E$ instead of $(\mathcal{A}_u, \mathcal{A}_v) \in E$ for compactness. Every $\overline{E}$ uniquely determines the structure of a Tree Augmented Naive Bayes classifier, because from $\overline{E}$ we can construct $\overline{E}^* = \overline{E} \cup \{(\mathcal{C}, \mathcal{A}_i) | 1 \leq i \leq n\}$ as can be seen in an example in Figure 3.1. We note $\overline{\mathcal{E}}^* = \{\overline{E}^* | \overline{E} \in \overline{\mathcal{E}}\}$. We note the root of a directed tree $\overline{E}$ as $\rho_{\overline{E}}$ (i.e. in Figure 3.1(b) we have that $\rho_{\overline{E}} = A_1$).

We will note as $\Theta_{\overline{E}^*}$ the set of parameters that quantify the Bayesian network $M = \langle \overline{E}^*, \Theta_{\overline{E}^*} \rangle$. More concretely:

$\Theta_{\overline{E}^*} = (\boldsymbol{\theta}_C, \boldsymbol{\theta}_{\rho_{\overline{E}}|C}, \{\boldsymbol{\theta}_{v|u,C} | u, v \in \overline{E}\})$

$\boldsymbol{\theta}_C = \{\theta_C(c) | c \in C\}$ where $\theta_C(c) = P(\mathcal{C} = c | M)$

$\boldsymbol{\theta}_{\rho_{\overline{E}}|C} = \{\theta_{\rho_{\overline{E}}|C}(i, c) | i \in A_{\rho_{\overline{E}}}, c \in C\}$ where
$\theta_{\rho_{\overline{E}}|C}(i, c) = P(\mathcal{A}_{\rho_{\overline{E}}} = i | \mathcal{C} = c, M)$

For each $u, v \in \overline{E}$: $\boldsymbol{\theta}_{v|u,C} = \{\theta_{v|u,C}(j, i, c) | j \in A_v, i \in A_u, c \in C\}$ where
$\theta_{v|u,C}(j, i, c) = P(\mathcal{A}_v = j | \mathcal{A}_u = i, \mathcal{C} = c, M)$.

3.4.1 Learning maximum likelihood TAN

One of the measures used to learn Bayesian networks is the *log likelihood*:

$$LL(B|\mathcal{D}) = \sum_{x \in \mathcal{D}} \log(P_B(x)) \tag{3.7}$$

An interesting property of the TAN family is that we have an efficient procedure (Friedman et al., 1997) for identifying the structure of the network which maximizes likelihood. The procedure and the theorem are given below where f is the probability distribution induced by the frequencies in $\mathcal{D}$.

Theorem 1 (Friedman, Geiger & Goldszmidt, 1997) *Let $\mathcal{D}$ be a dataset over Ω_C. The procedure* `Construct-TAN(`f`)` *builds a TAN B_T that maximizes $LL(B_T|D)$ and has time complexity $O(N \cdot n^2)$.*

To learn the maximum likelihood TAN we should use Theorem 1 to determine the structure and the following equation to compute the parameters.

$$\theta_{i|\Pi_i}(x_i, \Pi_{x_i}) = \frac{N_{i,\Pi_i}(x_i, \Pi_{x_i})}{N_{\Pi_i}(\Pi_{x_i})} \tag{3.8}$$

```
procedure Construct-TAN (ProbabilityDistribution P)
    var
        WeightMatrix I_P;
        UndirectedGraph UG;
        UndirectedTree UT;
        DirectedTree T;
        DirectedGraph TAN;
    foreach A_i, A_j
        Compute I_P(A_i; A_j|C) = sum_{x∈A_i, y∈A_j, z∈C} P(x,y,z) log( P(x,y|z) / (P(x|z)P(y|z)) )
    end
    G = ConstructUndirectedGraph(I_P);
    UT = MaximumWeightedSpanningTree(G);
    T = MakeDirected(UT);
    TAN = AddClass(T);
    return TAN;
```

Algorithm 1: TAN construction procedure

It has been shown (Friedman et al., 1997) that equation 3.8 leads to "overfitting" the model. Also in (Friedman et al., 1997) Friedman et al. propose to use the parameters as given by

$$\theta_{i|\Pi_i}(x_i, \Pi_{x_i}) = \frac{N_{i,\Pi_i}(x_i, \Pi_{x_i})}{N_{\Pi_i}(\Pi_{x_i}) + N^0_{i|\Pi_i}} + \frac{N^0_{i|\Pi_i}}{N_{\Pi_i}(\Pi_{x_i}) + N^0_{i|\Pi_i}} \frac{N_i(x_i)}{N} \tag{3.9}$$

and suggest setting $N^0_{i|\Pi_i} = 5$ based on empirical results. Using equation 3.9 to fix the parameters improves the accuracy of the classifier. In our opinion, no well founded justification is given for the improvement.

3.5 Bayesian model averaging for classification

We are faced with the problem of defining a good classifier for a classified dataset. If we accept that there is a probabilistic model behind the dataset, we have two alternatives:

1. We know the model M (both structure and parameters) that is generating the data in advance. In this case it is a matter of probabilistic computation. We should be able to calculate $P(\mathcal{C} = s_C | \mathcal{V} = S, M)$ for every $s_C \in C$ and to choose the class s_C with the highest probability. No learning is performed, because we knew the model in advance.

2. We are given a set of possible models $\mathcal{M}$. This is the situation, for instance, when learning decision trees, neural networks or Bayesian network

classifiers. The usual approach followed is to let an algorithm choose the model M that fits the data best. This can give good results, if the model selected accounts for a good share of the posterior probability distribution function over the set of models or if its predictions coincide with the ones given by the majority of the models. In spite of that, in this situation probability theory tell us we should take a weighted average where each model prediction is weighted by the probability of the model given the data. More formally, assuming ξ represents the hypothesis that the model underlying the data is known to be in $\mathcal{M}$ we have that:

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\mathcal{D}, \xi) = \int_{M \in \mathcal{M}} P(\mathcal{V} = S, \mathcal{C} = s_C|M) P(M|\mathcal{D}, \xi) \quad (3.10)$$

Applying this equation is commonly known as Bayesian model averaging (Hoeting et al., 1998). In practice, the problem is that for most models it is very hard to find a closed form for the integral. This has led to the introduction of methods such as Local Bayesian Model Averaging (LBMA) that approximate the integral over a subset of highly probable models.

3.6 Summary

In this chapter we have introduced the notation and terminology to be used in the rest of the thesis. We have also shortly reviewed naive Bayes, TAN and Bayesian model averaging. In chapter 5 we will see how to improve the interpretability of naive Bayes results. In chapters 6, 7, 8 and 9 we will see how the combination of the naive Bayes and TAN models with Bayesian model averaging and the principle of indifference introduced in Chapter 2 can be used to improve existing Bayesian network classifiers.

Chapter 4

A parallelizable distance-based discretization method

DISTANCE, n. The only thing that the rich are willing for the poor to call theirs, and keep.

Ambrose Bierce, The Devil's Dictionary

Discrete Bayesian network classifiers, as some other discrete classifiers, are not capable of dealing with continuous attributes. In this chapter we introduce an algorithm for distance based discretization that converts numerical attributes into ordered discrete ones. The technique we propose is competitive in accuracy with the best technique up to now and has the advantage of being easily parallelizable.

4.1 Introduction

Many approaches to classification assume the examples can be described only with discrete attributes. Others, like decision tree induction, increase its complexity when dealing with continuous attributes. Unfortunately, many real datasets include highly significant information in the form of continuous attributes. We have to take full advantage of this information without slowing learning.

Discretization is a process that transforms continuous attributes into discrete ones. Performing this previous step, we can apply discrete classification methods to datasets containing continuous values.

In this chapter we introduce a discretization method and show that it is parallelizable and that it achieves a top performance. The chapter begins introducing how discretization methods can be classified. After that, in section 4.3 we review some of the previous work done in discretization. After that we explain our algorithm and perform its parallelization in section 4.4. Section 4.5 gives the results of a set of empirical comparisons between the different discretization methods. We finish giving a set of conclusions in section 4.6.

4.2 Discretization methods classification

In (Dougherty et al., 1995) three different axis are used to make a classification of discretization methods. We will add two more, while keeping their three. A discretization method can be classified as either:

Global or Local Global methods divide each attribute in a number of regions that are the same for the whole dataset. Local methods face the problem locally, as for instance does C4.5 (Quinlan, 1992), that can set two different cutpoints for two different branches of the tree, generating two different discretization for two different subsets of the dataset. All the methods we will discuss are global

Supervised or Unsupervised A discretization method for classification can make use of the instance classes (supervised methods) to improve the quality of its discretization, or ignore this information (unsupervised methods or class-blind methods).

Static or Dynamic Static methods are those that decide each attribute discretization independently from each other. Dynamic methods try to discretize all the continuous attributes in the dataset simultaneously and this allows them to capture inter-attribute dependency, creating a more accurate discretization. Dynamic discretization has been less studied, and good discretization methods are expected to arise from this approach. All the discretization methods explained below are static ones.

Direct or Incremental Direct methods divide the range in k intervals simultaneously, needing an additional criterion to determine the value of k.

Incremental methods begin with a simple discretization and pass though a improvement process, needing an additional criterion to know when to stop the discretization.

Bottom-Up or Top-Down Incremental methods usually can be divided into Top-Down and Bottom-Up. Top-Down methods begin with an empty discretization and its improvement process is simply to add a new cutpoint to the discretization. Bottom-Up methods begin with a discretization that has all the possible cutpoints and its improvement process consists in merging two intervals (delete a cutpoint).

4.3 Some discretization methods

In this section we shortly review some of the existent discretization methods.

4.3.1 Equal size

The simplest discretization method is an unsupervised direct method named *equal size discretization*. It calculates the maximum and the minimum for the attribute that is being discretized and partitions the range observed into k equal sized intervals.

4.3.2 Equal frequency

This is another unsupervised direct method. It counts the number of values we have from the attribute that we are trying to discretize and partitions it into intervals containing the same number of examples.

These two methods need a value (say k) for the number of intervals. Experimentally it has been shown that taking $k = max\{1, log(l)\}$, where l is the number of different values of the attribute is a good heuristic.

4.3.3 ChiMerge

ChiMerge is a supervised, incremental, bottom-up method designed by Randy Kerber and described in (Kerber, 1992). Kerber defines a criterion for measuring the quality of a discretization, stating that *"in an accurate discretization, the relative class frequencies should be fairly consistent within an interval but two adjacent intervals should not have similar relative class frequencies"*. ChiMerge uses χ^2 statistic to determine the *independence* of the class from the two adjacent intervals, combining them if it is *independent*, and allowing them to be separate otherwise.

The ChiMerge algorithm has two main stages. In the first it sorts the examples by the attribute to be discretized and initializes the discretization putting each instance in its own interval.

The second stage is a bottom-up merging process, that contains two steps that are repeated continuously until no more merges are needed. In the first

step the χ^2 value is computed for each pair of adjacent intervals by means of equation 4.1. In the second the pair with the lowest χ^2 value is merged. We say that no more merges are needed when all the χ^2 values are above some χ^2 threshold. The χ^2 threshold is determined by selecting a desired significance level and then using a table or formula to obtain the corresponding χ^2 value.

If we refer to the discrete attribute we are constructing as R, the χ^2 value is calculated with the following formula:

$$\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{\#C} \frac{(N_{R,C}(i,j) - E(i,j))^2}{E(i,j)} \tag{4.1}$$

where

$\#C$ is the number of different classes

$N_{R,C}(i,j)$ is the number of examples in the ith interval, jth class

$E(i,j) = \frac{N_R(i)N_C(j)}{N}$ is the expected value of $N_{R,C}(i,j)$

$N_R(i)$ is the number of examples in the ith interval

$N_C(j)$ is number of examples in the jth class

N is total number of examples

The computational complexity of the method when some optimizations are done is $\mathcal{O}(N \cdot \log N)$ where N is the number of examples of the dataset being discretized.

4.3.4 Entropy

Entropy discretization is a supervised, incremental, top-down method proposed by Usama M. Fayyad and Keki B. Irani and described in (Fayyad and Irani, 1992) and (Fayyad and Irani, 1993). Entropy discretization recursively selects the cutpoints minimizing entropy until a stopping criterion based on the Minimum Description Length criterion ends the recursion.

Entropy discretization starts by sorting all the examples by the attribute being discretized. Once sorted, it uses a recursive divide and conquer approach to discretization. A sketch of the algorithm is presented in Algorithm 2.

In the first step it evaluates each possible cutpoint, and selects the one which induced partition has the minimal class information entropy. We need some definitions to clarify the previous statement. Let there be $\#C$ classes $C_1,\ldots,C_{\#C}$, and let $f_C^{\mathcal{D}}(i)$ be the proportion of examples in $\mathcal{D}$ that have class C_i. The class entropy of a dataset $\mathcal{D}$ is defined as:

$$Ent(\mathcal{D}) = -\sum_{i=1}^{\#C} f_C^{\mathcal{D}}(i) \log f_C^{\mathcal{D}}(i) \tag{4.2}$$

```
function EntDisc(Dataset D, Attribute A)
begin
    Choose the best cutpoint (T_A)
        according to the entropy criterion
    Evaluate if the cutpoint is significant
        according to the Minimum Description Length Principle.
    if it is not significant then
        /* no further discretization is needed */
        return the empty list.
    else
        D_1 = {x ∈ D | x_A < T_A}
        D_2 = {x ∈ D | x_A > T_A}
        return EntDisc(D_1,A),T_A,EntDisc(D_2,A).
    endif
end
```

Algorithm 2: Entropy discretization method

Given a dataset $\mathcal{D}$, an attribute A, a cutpoint value T, and assuming that T partitions $\mathcal{D}$ into $\mathcal{D}_1$ and $\mathcal{D}_2$, the class information entropy is calculated by the expression:

$$E(A,T;\mathcal{D}) = \frac{N^{\mathcal{D}_1}}{N^{\mathcal{D}}} Ent(\mathcal{D}_1) + \frac{N^{\mathcal{D}_2}}{N^{\mathcal{D}}} Ent(\mathcal{D}_2) \tag{4.3}$$

where $N^{\mathcal{D}}$,$N^{\mathcal{D}_1}$,$N^{\mathcal{D}_2}$ are the number of instances in $\mathcal{D}$, $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively. What entropy discretization does is selecting the cutpoint T_A for which $E(A,T_A;\mathcal{D})$ is minimal amongst all the candidate cutpoints.

In the second step the Minimum Description Length Principle (MDLP from now on) is used to evaluate whether a cutpoint is significant or not. Now we introduce the MDLP and after that we will see how it can be applied to our problem.

Statement 1 (Minimum Description Length Principle) *Given a set of competing hypotheses and a set of data $\mathcal{D}$, the MDLP calls for selecting the hypothesis HT for which $MLength(HT) + MLength(\mathcal{D}|HT)$ is minimal among the set of hypothesis. $MLength(HT)$ denotes the length of the minimal possible encoding of HT, while $MLength(\mathcal{D}|HT)$ is the length of the minimal encoding of the data given the hypothesis.*

In our case we will have two hypothesis. The first one, is that no further discretization is needed, and the second one is that it is better to put a cutpoint in T_A. Analyzing the minimal encoding length that each of this hypothesis generate we arrive to a decision criterion (the complete development appears in (Fayyad and Irani, 1993))

The partition induced by a cutpoint T for a set S of N examples is accepted if and only if

$$Gain(A,T;\mathcal{D}) > \frac{\log_2(N-1)}{N} + \frac{\Delta(A,T;\mathcal{D})}{N} \tag{4.4}$$

where

$$Gain(A,T;\mathcal{D}) = Ent(\mathcal{D}) - E(A,T,\mathcal{D}) \tag{4.5}$$

$$\Delta(A,T;\mathcal{D}) = \log_2(3^{\#C}-2) - [\#C Ent(\mathcal{D}) - \#C^{\mathcal{D}_1} Ent(\mathcal{D}_1) - \#C^{\mathcal{D}_2} Ent(\mathcal{D}_2)] \tag{4.6}$$

where $\#C$ is the number of different classes in $\mathcal{D}$ and $\#C^{\mathcal{D}_1}$,$\#C^{\mathcal{D}_2}$ the number of classes present in $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively.

The computational complexity of the algorithm is bounded by the sort stage and is $\mathcal{O}(N \cdot \log N)$, where N is the number of examples in our dataset.

4.3.5 D-2

D-2 is a supervised, incremental, top-down method proposed by J. Catlett and described in (Catlett, 1991). D-2 recursively selects the cutpoints maximizing Quinlan's Gain until a stopping criterion based on a set of heuristic rules ends the recursion. Its main algorithm appears in Algorithm 3.

```
function D2Disc(Dataset D, Attribute A)
begin
    Choose the best cutpoint (T_A)
        according to the gain criterion
    Evaluate if the cutpoint is significant
        according to the D-2 heuristics.
    if it is not significant then
        /* no further discretization is needed */
        return the empty list.
    else
        D_1 = {x ∈ D | x_A < T_A}
        D_2 = {x ∈ D | x_A > T_A}
        return D2Disc(D_1, A), T_A, D2Disc(D_2, A).
    endif
end
```

Algorithm 3: D-2 discretization method

As can be seen in the algorithm, D-2 selects at each step the cutpoint that maximizes the information gain.

The stopping criterion is composed by a set of sub-conditions that must be accomplished for accepting a cutpoint. These conditions are:

- The number of examples in the interval must be greater than 14.

- The maximum number of thresholds is seven.
- The gain on all possible thresholds must not be equal.
- The examples in the interval must pertain to different classes.

These heuristics are explained more deeply in (Catlett, 1991).

The computational complexity of the method is bounded by the sort stage, and hence is $\mathcal{O}(N \cdot \log N)$ where N is the number of examples in the dataset being discretized.

4.3.6 Other discretization methods

There are some more discretization methods in the literature. A good comparison between some of them is found in (Dougherty et al., 1995). A discretization method based in Hellinger divergence is introduced in (Lee and Shin, 1994). A statistical method named StatDisc is described in (Richeldi and Rossotto, 1995). A method for describing the accuracy of discretization methods for Bayesian network classifiers appears in (Pazzani, 1995). Finally, in (Friedman and Goldszmidt, 1996) a method for discretizing continuous attributes while learning Bayesian networks is presented.

4.4 Distance-Based discretization method

In this section we describe a new discretization algorithm based on a distance between partitions (López de Màntaras, 1991). The algorithm is global, supervised, static and Top-Down incremental. This means that it is required to have two main components, a cutpoint selection criterion and a stopping criterion. The algorithm has a first step that orders the examples by the attribute value. Once the examples have been sorted, the main loop of our implementation of the method appears in Algorithm 4.

```
function MDisc(Dataset 𝒟,Attribute A)
begin
    Disc = ∅
    NewCutPoint = SelectNewCutPoint(𝒟,A,Disc)
    while (ImprovesDiscretization(𝒟,A,Disc,NewCutPoint))
        Disc = Disc ∪{ NewCutPoint }
        NewCutPoint = SelectNewCutPoint(𝒟,A,Disc)
    end
    return Disc
end
```

Algorithm 4: Distance-Based discretization

There is a main difference between our algorithm structure and previously seen Top-Down incremental algorithms structure: D-2 and Entropy were recursive, dividing the set in two subsets and recursively discretizing each subset following a divide and conquer strategy while our algorithm is iterative, considering the whole set for the selection of each new cutpoint.

4.4.1 Cutpoint selection criterion

In decision tree induction algorithms, such as the ID3 algorithm, we estimate the classification power of an attribute by some measure such as Gain, Gain Ratio or 1 - Distance. We want to find a set of cutpoints so that the classification power of the resulting discretized attribute is the highest possible. Our idea is to follow a greedy heuristic in this search.

Each discretization can be identified with a set of cutpoints. We denote by $\mathcal{P}_{Disc}$ the partition induced by a discretization. We will use $\mathcal{P}_{Disc\cup\{T\}}$ for noting the partition induced in our dataset when the discretization applied to our attribute is the result of adding cutpoint T to the discretization $Disc$. We impose that the function SelectNewCutPoint has to find a cutpoint T_A that accomplishes:

$$\forall T, Dist(\mathcal{P}_C, \mathcal{P}_{Disc\cup\{T\}}) > Dist(\mathcal{P}_C, \mathcal{P}_{Disc\cup\{T_A\}}) \tag{4.7}$$

where $\mathcal{P}_C$ is the partition generated in the dataset by the class attribute and $Dist$ stands for the distance, which is defined as:

$$Dist(\mathcal{P}_C, \mathcal{P}_{Disc}) = \frac{Inf(\mathcal{P}_C|\mathcal{P}_{Disc}) + Inf(\mathcal{P}_{Disc}|\mathcal{P}_C)}{Inf(\mathcal{P}_C \cap \mathcal{P}_{Disc})} \tag{4.8}$$

where

$$Inf(\mathcal{P}_C|\mathcal{P}_{Disc}) = Inf(\mathcal{P}_C \cap \mathcal{P}_{Disc}) - Inf(\mathcal{P}_{Disc}) \tag{4.9}$$

$$Inf(\mathcal{P}_C \cap \mathcal{P}_{Disc}) = -\sum_{i=1}^{\#C}\sum_{j=1}^{\#Disc} Prob(C_i \cap Disc_j)\log_2\left(Prob(C_i \cap Disc_j)\right) \tag{4.10}$$

$$Inf(\mathcal{P}_{Disc}) = -\sum_{i=1}^{\#Disc} Prob(Disc_i)\log_2\left(Prob(Disc_i)\right) \tag{4.11}$$

are the standard Shannon measures of information of a partition. The probabilities are approximated by the frequencies in the dataset. For more details see (López de Màntaras, 1991; López de Màntaras et al., 1998).

In the first run of the function SelectNewCutPoint, a contingency table is calculated for each cutpoint, storing the class distribution on each side of it. If it is not the first run, we modify each table by adding the new cutpoint. After that we evaluate the distance between the partition generated by the discretization with the new cutpoint and the partition generated by the class. We do this for

each possible new cutpoint, storing the minimum distance and the cutpoint to which it corresponds. When every possible cutpoints has been processed, the one with minimum distance is returned.

Once T_A is found, the next step is Algorithm 4 if the cutpoint improvement is significant enough to accept it or if the improvement is insignificant and no further cutpoints are considered necessary for the discretization.

4.4.2 The stopping criterion

We need a heuristic to evaluate improvement. Our first approach to this problem was similar to the set of heuristics used in D-2. The results with this approach were acceptable, but after a deeper analysis we realized that for some datasets the discretization was unnecessarily complex and for others some useful cutpoints were not included in it. Therefore we decided to develop a stopping criterion based on the Minimum Description Length Principle (MDLP), like the one used by Entropy discretization.

The development is parallel to the one in (Fayyad and Irani, 1993). The problem that needs to be solved is a communication problem. We have to communicate a classifier method, that allows the receiver to determine the class of each example. The sender knows all the attributes of the examples, plus the class, and the receiver knows all the attributes of the examples but not the class. The sender must choose the shortest description for sending a message that allows the receiver to correctly classify each example. Suppose we have a discretization with p cutpoints, and we are analyzing whether we want to include the $p+1$ cutpoint or not. Our sender has essentially two possible descriptions, one without the new cutpoint and the other with it. We calculate the encoding length of each description and choose the option which is minimal.

The encoding length of communicating the set of classes based on a p-cutpoint discretization can be decomposed as:

$$Length(Disc) + Length(\mathcal{D}|Disc) \tag{4.12}$$

The first term of 4.12 is

$$Length(Disc) = p\ \log(N-1) + (p+1)\#C + \sum_{i=0}^{p} \#C^{\mathcal{D}_i} Ent(\mathcal{D}_i) \tag{4.13}$$

where $\mathcal{D}_i$ is the subset of the dataset containing the instances with a value of the discretized attribute in the interval i, $\#C^{\mathcal{D}_i}$ is the number of different classes in $\mathcal{D}_i$ and $Ent(\mathcal{D}_i)$ is the Shannon entropy for the partition induced by the class in $\mathcal{D}_i$.

The first term of 4.13 is the length needed to make the receiver know where the p cutpoints are (we communicate the position of each cutpoint in the ordered sequence of examples). The second term encodes which classes are present in each interval. It is necessary, because the receiver needs this information to decode the third term of the expression that defines a code book for each interval

in the discretization. The length of the code book is given by the following theorem (Hamming, 1980):

Theorem 2 *Given a source of messages s with entropy $Ent(s)$, for any $\varepsilon > 0$, there exists an optimal encoding of the messages of s such that the average message code length l, in bits, is such that $Ent(s) \leq l \leq Ent(s) + \varepsilon$*

Once we have communicated the discretization, we need to communicate the classes. As we have communicated a codebook, we can calculate the length of communicating the classes in $\mathcal{D}$ by means of theorem 2:

$$Length(\mathcal{D}|Disc) = \sum_{i=0}^{p} N^{\mathcal{D}_i} Ent(\mathcal{D}_i) \tag{4.14}$$

We have characterized expression 4.12 completely. Given two discretizations, one with p and the other with $p+1$ cutpoints, we will choose that with the minimal encoding length. If it is the one with p cutpoints, then we stop our algorithm and no more cutpoints are added to the discretization. Otherwise we consider including another cutpoint.

4.4.3 Computational complexity

The computational complexity of the method is not easily measurable, because the stopping criterion depends on the data in which we are working. The complexity of the sorting step is $\mathcal{O}(N \cdot \log N)$. The complexity of the function SelectNewCutPoint in our implementation (it reuses the results of the previous runs) is $\mathcal{O}(\#C \cdot i \cdot N)$ where $\#C$ is the number of classes in the dataset, N is the number of examples in the dataset and i is the number of intervals the discretization has in this run. The complexity of ImprovesDiscretization is $\mathcal{O}(\#C \cdot i)$. We will not consider it, because $\mathcal{O}(\#C \cdot i) \subset \mathcal{O}(\#C \cdot i \cdot N)$. Suppose we discretize the attribute with p cutpoints. The total complexity of the method is given by:

$$\mathcal{O}\left(N \cdot \log(N) + \sum_{i=1}^{p} \#C \cdot N \cdot i\right) \tag{4.15}$$

that simplifying gives:

$$\mathcal{O}((log(N) + p^2 \cdot \#C) \cdot N) \tag{4.16}$$

k is constant, and very small with respect to N. To ease the evaluation of the complexity, we can use a heuristic restriction as the one imposed in D-2, and say that discretizations cannot have more than 7 intervals. With this assumptions, complexity is bounded by the sorting step, as for Entropy, D-2 or ChiMerge.

4.4.4 Parallelization of the method

In the previous point, analyzing computational complexity, we have found that the complexity of the algorithm, without including the sorting step, is mainly

related with the complexity of the function SelectNewCutPoint. If we are able to parallelize this function, it will give us a high improvement in the performance of the algorithm.

Suppose we assign a processor to each example in the dataset. Then the parallelized version of the algorithm is as follows:

- Step 1. The sorting step can be parallelized with N processors in time $\mathcal{O}(\log N)$. From now on we assume the values are sorted in ascending order by the attribute being discretized.
- Step 2. We have to calculate a contingency table for each processor, in order for the processor to be able to evaluate the Distance between the partition generated by the class and the partition generated by fixing the cutpoint just between its value and the value of the neighbour processor on its left side. This can be done in $\mathcal{O}(\#C \cdot \log N)$ time in two steps, the first one by adding the information of all the processors following a processor binary tree until it arrives to the root, and the second one by descending the processors tree, distributing the information we have previously put together in the first step.
- Step 3. Now we have that each processor has its corresponding contingency table. Each processor evaluates independently the Distance measure for its cutpoint. This is done in time $\mathcal{O}(\#C \cdot i)$ where i is the number of cutpoints that have been added to the discretization up to now.
- Step 4. We have to calculate the processor with minimal Distance. We use again the binary processor tree, and this gives us a time $\mathcal{O}(\log N)$.
- Step 5. The root processor evaluates the MDL criterion to see if the new cutpoint must be added to the discretization. If it turns out that the new cutpoint is not good enough, broadcasts it to the other processors, and the algorithm stops here. Otherwise the new cutpoint is annotated by the root processor. The information of where the cutpoint has been fixed, and the contingency table of the processor which cutpoint has been selected is broadcasted to all the processors. This can be bounded in time by $\mathcal{O}(\#C \cdot i \cdot \log N)$
- Step 6. Each processor transforms its contingency table by taking into account that a new cutpoint has been fixed, using its old contingency table, and the table broadcasted in the previous step. This step is bounded by $\mathcal{O}(\#C \cdot i)$
- Step 7. We return to step 3.

Suppose as in section 4.4.3 that the attribute is discretized with p cutpoints. The complexity of the sequence of steps is bounded by time $\mathcal{O}(\#C \cdot p^2 \cdot \log N)$ and $\mathcal{O}(N)$ processors. A parallel version of the method following this approach has been implemented in MPI and the main part of its code can be examined in (Cerquides, 1997). Parallelization of other methods, such as Entropy or D-2

is not direct, due to their recursive nature. Obviously this does not mean that they are not parallelizable, only that, since they are based on recursion splitting the dataset, no direct simple parallelization can be applied to them.

4.5 Empirical comparison

4.5.1 Comparison design

There is no way of directly evaluating a discretization, so we will use the accuracy of two classification algorithms to measure the discretization goodness. The two algorithms will be ID3 (Quinlan, 1986) (with no pruning) and Naive-Bayes (Langley et al., 1992). We will run each algorithm in 9 different domains with different characteristics (see Table 4.1). For each learning algorithm, discretization method and dataset we do 50 runs, each one with 70% of the examples as training set and the remainder 30% as test set. We take the average of the 50 runs as a measure of performance. We also keep the results of the 50 runs to make two statistical significance tests: Rank and Signed Rank. In (Cerquides, 1997),(Gibbons, 1971),(López de Màntaras et al., 1998) we can find a complete explanation of these tests.

Dataset	Attributes	Instances	Classes	Missing
CRX	15	690	2	few
ECHO	7	131	2	some
GLASS	10	214	7	none
HEARTC	13	303	2	several
HEARTH	13	294	2	some
HEP	19	155	2	some
HORSE	27	368	2	30 %
IRIS	4	150	3	none
WINE	13	178	3	none

Table 4.1: Domains used for the discretization methods comparison

4.5.2 Comparison results

Average accuracies comparison

For each dataset and classification algorithm we rank the 6 discretization methods, from the first place (the most accurate) to the sixth one. The results are displayed in the two tables 4.2 and 4.3. The rows are ordered with the best method on the top and the worst on the bottom. In the tables 55555 means the algorithm ranked five times in the position specified by the column under which it appears, 4444 four times, 333 three times and so on.

We can extract some conclusions from this two tables:

	First	Second	Third	Fourth	Fifth	Sixth
DISTANCE	55555	4444				
ENTROPY	4444	55555				
SIZE			55555	333	1	
D2			22	4444	22	1
CHIMERGE			22	1	1	55555
FREQUENCY				1	55555	333

Table 4.2: Average accuracy results for the different discretization methods using ID3

	First	Second	Third	Fourth	Fifth	Sixth
DISTANCE	55555	22		22		
ENTROPY	1	333	22	1	1	1
FREQUENCY	22		333	1	22	1
SIZE	1	333		22	1	22
D2		22	333	1	22	1
CHIMERGE				22	333	4444

Table 4.3: Average accuracy results for the different discretization methods using Naive-Bayes

- Distance seems to be globally the best performer, while ChiMerge seems to be the worst.
- For the ID3 classification algorithm, either Distance or Entropy have always the first place.
- Frequency performs badly with ID3, but considerably well with Naive-Bayes.

Significance test comparison

For each dataset and each classification algorithm we have performed the pairwise comparison of accuracies for the six discretization methods. This has given us a partial ordering of the methods for each dataset. The complete results appear in (Cerquides, 1997). We will try to extract the most important conclusions in a few statements:

- Rank and Signed Rank tests differ only in one over twenty of the comparisons, and when they do, it never results in a sign change. Therefore we have decided to analyze only Signed Rank results.
- Entropy and Distance are better than the rest in a statistically significant way for all the datasets when using ID3 as classification algorithm.

- For the Naive Bayes classifier, Distance seems to perform better than the rest, but in most of the cases it is not statistically significantly better than Entropy.

4.6 Summary

We have introduced a new method for discretization of continuous values. We have seen that our new method has an acceptable time complexity. We have also shown that our method is easily parallelizable, and we have implemented a parallel version of it. This characteristic is specially important for its use in Knowledge Discovery in Databases (KDD), because databases in that area are supposed to have a large number of registers. The time complexity constraints of algorithms used in the KDD area are very strong, and parallel computation is a good way for reducing it.

We have also compared our discretization method, in terms of accuracy, with other five methods, and for two different classification algorithms (ID3 and Naive-Bayes) observing that it has better average accuracy than the best of the methods proposed until now (the Entropy method), but this difference is not statistically significant.

We have shown that our method can be a good alternative to Entropy discretization, especially for very large datasets, where a time complexity $\mathcal{O}(N \cdot \log N)$ may be unacceptable.

We think that discretization of continuous attributes has many open research issues. We believe that it would be very interesting to face the problem of discretization taking into account possible relationships between the attributes. In our analysis, we have avoided to take into consideration the amount of noise in the datasets. New discretization methods can be designed especially for working in noisy environments where errors or missing values are usual. The datasets we have used for comparing the performance of our method have at most several hundreds of elements. A new study must be done for larger datasets. In (Pazzani, 1995) a refinement process for discretizations is proposed. It will be interesting to study how this refinement affects our discretization method.

Chapter 5

The Qualitative Bayesian Classifier

It is quality rather than quantity that matters.

Seneca (5 BC - 65 AD), Epistles

Bayesian reasoning has been usually criticized as hard to explain and understand, but achieves high performance rates with simple constructs, as happens for instance with the naive Bayes classifier (Langley et al., 1992). In this chapter we review some approaches to qualitative uncertainty and propose a new one based on the idea of Absolute Orders of Magnitude. The approach is tested by implementing a classification method based on this idea. This classification method is an adaptation of the naive Bayes classifier, that keeps its accuracy while gaining interpretability.

5.1 Introduction

Comprehensibility is a key characteristic for machine learning models to be useful in data mining tasks. Some approaches to increasing Bayesian reasoning comprehensibility appear in (Johnson, 1973), (Lichtenstein and Newman, 1967), (Wallsten et al., 1985), (Zimmer, 1983) and (Zimmer, 1985). The main idea in all of them is to attach linguistic labels as "probable" or "very unlikely" to numerical probabilities, that is to absolutes degrees of belief. Bayesian reasoning works primarily with changes in probability values, and these approaches do not seem to give any interpretation of such changes, giving as result hardly understandable explanations. It has been accepted that, unlike physical parameters, absolute probabilities do not seem to have values (except the endpoints) that are universally interesting (Wellman, 1990).

This problem was noticed also by Elsaesser, that in (Elsaesser, 1989) proposed the use of a version of Polya's "shaded inductive patterns" (Polya, 1954) for linguistic explanation of Bayesian inference. Elsaesser uses Oden's model (Oden, 1977) to create the linguistic labels related to changes in probability. Elsaesser explanations are comprehensible, but we have no security that reasoning with the information given by these explanations really bring us to coherent conclusions because explanation and reasoning are performed at different levels, and we are not allowed to use a previous explanation in a future case.

Another approach is the one followed by Neufeld (Neufeld, 1990), Wellman (Wellman, 1990) and Parsons (Parsons, 1995), using ideas from the field of qualitative uncertainty. The idea behind their work is finding whether a fact A is favoured, unfavoured or not altered by another fact B. Quoting Parsons:

> Whereas in probabilistic networks the main goal is to establish probabilities of hypotheses when particular observations are made in qualitative systems the main aim is to establish how values change.

Our approach can be viewed as a refinement of qualitative probabilistic networks (QPNs) showing that, slightly modified, Elsaesser explanations can be used not just for explanatory purposes but also for reasoning and prediction achieving results similar to those of non-qualitative probabilistic reasoning, while keeping intact its interpretability.

Next section briefly introduces qualitative probabilistic networks, concretely Wellman and Neufeld approaches. Section 5.3 introduces our qualitative approach to influences and synergies, making use of the absolute orders of magnitude model. Section 5.4 describes our proposal to use the qualitative influences and synergies in a Qualitative Bayesian classifier. Section 5.5 describes an empirical comparative study based on 15 datasets and analyses the results obtained with the aim of showing the good performance of our approach in terms of classification accuracy. In section 5.6 we use examples from one of the datasets to show the explanation and description capabilities of the proposed approach. We end up with a short summary in section 5.7.

5.2 Introduction to Qualitative Probabilistic Networks

The concept of QPN has been approached by two ways. We will shortly review both here.

5.2.1 Wellman approach

For Wellman, a QPN is a pair $G = (V, Q)$, where V is the set of variables or vertices of the graph and Q is a set of qualitative relationships among the variables. He introduces two main concepts for modeling QPNs as are *qualitative influences* and *qualitative synergies*.

Wellman qualitative influences

Qualitative influences can be thought of as qualitative relations describing the sign (direction) of the relationship between a pair of variables. A variable can influence another positively (+), negatively (-), or in no way (0). We should also consider the possibility that the sign of the influence is unknown to us (?). If we use δ to denote one of {+,-,0,?} we say a qualitative influence of a on b in direction δ holds in the graph $G = (V, Q)$ if $(a, b, \delta) \in Q$. For formally introducing the probabilistic semantic of this concept the way Wellman does, we need to previously define the set of predecessors that influence a variable in a network.

$$pred_G(b) = \{a | (a, b, \delta) \in Q, for some\ \delta \in \{+, -, ?\}\}$$

Now we can assign probabilistic meaning to influences. We say that an influence edge $(a, b, +) \in Q$ is satisfied in a concrete domain if for all $x \in pred_G(b) - \{a\}$ such that x is consistent with both a and $\neg a$, we have

$$Pr(b|a, x) \geq Pr(b|\neg a, x)$$

The meaning of this expression can be stated as: under any circumstances (x) that are known to affect b, the presence of a makes b more likely than its absence.

Parallel definitions can be done for $(a, b, -)$, $(a, b, 0)$ and $(a, b, ?)$, replacing $\geq$ by $\leq$, $=$, and "no condition at all" ($(a, b, ?)$ always holds) respectively.

Wellman qualitative synergies

Qualitative synergies describe the qualitative interaction among influences. The idea behind them is that two variables synergically influence a third if their joint influence is greater than separate, statistically independent, influences. The formalization of this idea can be seen in (Wellman, 1990), and will be skipped here.

5.2.2 Neufeld approach

Neufeld formalizes the idea of qualitative influence by means of the concept of favouring. He says a favours b if

$$Prob(b|a) > Prob(b)$$

He includes four types of edges in what he calls inference graphs:

- *Defeasible links.* Given a, b is more likely to happen.

  $a \rightarrow b$ if $1 > Prob(b|a) > Prob(b)$

- *Logical links.* Given a, b will surely happen.

  $a \Rightarrow b$ if $1 = Prob(b|a) > Prob(b)$

- *Negative defeasible links.* Given a, b is less likely to happen.

  $a \not\rightarrow b$ if $1 > Prob(\neg b|a) > Prob(\neg b)$

- *Negative logical links.* Given a, b will not happen.

  $a \not\Rightarrow b$ if $1 = Prob(\neg b|a) > Prob(\neg b)$

Once introduced these concepts Neufeld uses them to do common sense reasoning. For more details on his approach to qualitative uncertainty see (Neufeld, 1990).

5.3 Influences and synergies revisited

Neufeld and Wellman ideas are useful for common sense reasoning, planning under uncertainty and when qualitative differential equations are not applicable. Our idea is to adapt them in order to make them useful for classification and characterization of sets.

The qualitative model used by both approaches is the signs model, composed of three categories +,-,0 and ? for representing the unknown. More sophisticated models have risen from the field of qualitative reasoning. One of these models is the absolute orders of magnitude model (Trave and Piera, 1989), that considers a finer partition of the real line than the one given by the signs, allowing also distinctions in quantities of the same sign. This model qualifies quantities into seven classes, from *Negative Large* to *Positive Large*, including *Zero*. Quantities of the same sign are divided into three classes (*Large*, *Medium* and *Small*) that are very natural in human reasoning. We have discretized influences into this new model, in a way coherent with Neufeld works. Neufeld states $a \ favours \ b$ if $Prob(b|a) > Prob(b)$ that is, if $\frac{Prob(b|a)}{Prob(b)} > 1$. This quotient was also used by

Elsaesser, in his work trying to explain Bayesian reasoning, to denote the shift in belief that a produces in b. We will define influence of a in b as:

$$Influence(a, b) = \frac{Prob(b|a)}{Prob(b)} \tag{5.1}$$

We note that:

$$Influence(a, b) = Influence(b, a) \tag{5.2}$$

Once we have a definition for influence, we make use of the absolute orders of magnitude model to make influences comprehensible. By discretizing influences into the seven classes seen in Figure 5.1, we perform a process similar to that of Elsaesser assigning linguistic labels as "much more likely", "a little less likely", and so on. We tested different alternatives for the boundary values between labels. The boundary values established in Figure 5.1 were selected over a set of alternatives because they performed better in the classification experiments described in section 5.5. These boundary values did in fact perform better in these experiments that an alternative adaptive discretization schema where the boundaries were calculated by a discretization algorithm (concretely Entropy discretization (Fayyad and Irani, 1992; Fayyad and Irani, 1993)). This is possibly due to the fact that the rationale behind the values in Figure 5.1 is that they should be symmetric, in the sense that when multiplied the result should be one. This is not fulfilled by the boundary values provided by a discretization algorithm.

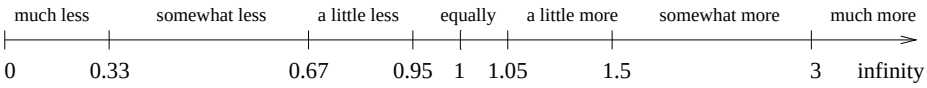


Figure 5.1: Influence discretization scale

We have given an expression for influences and a scale for their discretization. Synergies can be seen as the difference in influence between two facts that happen together with respect to these two facts happening separately. We can give the following expression for synergies of two variables:

$$Synergy(\{a_1, a_2\}, b) = \frac{Influence(a_1 \cap a_2, b)}{Influence(a_1, b) * Influence(a_2, b)} \tag{5.3}$$

5.4 The Qualitative Bayesian Classifier

In this section we will show that qualitative influences and synergies can be used for reasoning and, more concretely, for classification tasks, giving high classification rates. We will use them to get a qualitative version of the naive Bayes classifier. Naive Bayes has already been introduced in chapter 3. Concretely,

in section 3.3, we state that naive Bayes tells us to choose the class s_C which maximizes:

$$P(\mathcal{C} = s_C|M) * \prod_{i=1}^{n} P(\mathcal{A}_i = s_i|\mathcal{C} = s_C, M) \tag{5.4}$$

An interpretation of this formula can be that $P(\mathcal{C} = s_C|M)$ is our initial belief in the fact that s_C is the class of our example, and each one of the factors $P(\mathcal{A}_i = s_i|\mathcal{C} = s_C, M)$ can be seen as shifts that modify this belief with regard to the new information that $\mathcal{A}_i = s_i$. We will adapt these shifts in belief to coincide with our previously defined influences. In order to do this, we will need to add the assumption that the attributes are statistically independent. Given this assumption, we can modify equation 3.4, developing the denominator the same way we did with the numerator, that is

$$P(\mathcal{V} = S|M) = \prod_{i=1}^{n} P(\mathcal{A}_i = s_i|M) \tag{5.5}$$

Substituting in equation 3.4 we have:

$$P(\mathcal{C} = s_C|\mathcal{V} = S, M) = P(\mathcal{C} = s_C|M) * \frac{\prod_{i=1}^{n} P(\mathcal{A}_i = s_i|\mathcal{C} = s_C, M)}{\prod_{i=1}^{n} P(\mathcal{A}_i = s_i|M)} \tag{5.6}$$

If we refine the definition of influences appearing in equation 5.1 assuming a naive Bayes model M as

$$Influence(\mathcal{A}_i = s_i, \mathcal{C} = s_C) = \frac{P(\mathcal{A}_i = s_i|\mathcal{C} = s_C, M)}{P(\mathcal{A}_i = s_i|M)} \tag{5.7}$$

this allows us to express equation 5.6 as:

$$P(\mathcal{C} = s_C|\mathcal{V} = S, M) = P(\mathcal{C} = s_C|M) * \prod_{i=1}^{n} Influence(\mathcal{A}_i = s_i, \mathcal{C} = s_C) \tag{5.8}$$

Now we can apply this rule with qualitative influences and analyze the differences in accuracy between applying the naive Bayes classifier where shifts in belief vary continuously from 0 to 1 and our qualitative influences framework, where shifts only can have the seven values we have previously seen.

Before doing this, we introduce synergies in our classifier, because synergies can be seen as trying to express interattribute dependencies with respect to the class and hence including them can improve the performance when the classification problem at hand does not fulfill the independence assumptions. Our first idea was to calculate all the synergies between all pairs of variables and apply them. The problem with this approach is that it is not an approximation of the Bayes formula, and hence is not theoretically well founded and gives a poor empirical result. In fact, introducing synergies that way worsened accuracy. The

```
program SOQBC;
    foreach class c ∈ C
        InfEffect = ∏_{i=1}^{n} QInfluence(A_i = s_i, C = c)
        ClassP[c] = P(C = c|M) * InfEffect
        InfCorrected[c] = ∅;
        ApplySynergies(LargeSynergies(c),c);
        ApplySynergies(MediumSynergies(c),c);
        ApplySynergies(SmallSynergies(c),c);
    end
    Select the class c with highest ClassP[c];
end
procedure ApplySynergies(Set SynergySet, Class c)
    while SynergySet ≠ ∅
        Randomly choose a Synergy (namely {A_i = s_i, A_j = s_j})
            and delete it from SynergySet
        if ({A_i = s_i, A_j = s_j} ∩ InfCorrected[c] = ∅) then
          QSyn = QSynergy({A_i = s_i, A_j = s_j}, C = c)
          ClassP[c] = ClassP[c] * QSyn
          InfCorrected[c] = InfCorrected[c] ∪ {A_i = s_i, A_j = s_j}
```

Algorithm 5: Synergies application strategy

reason is that synergies can be seen as corrections of the approximation to the probabilities given by influences. It is not correct to apply a synergy correction for two variables A_i and A_j and also apply it to A_j and A_k, because we are correcting A_j influence twice. That is why we follow the next schema for applying synergy corrections:

> We first classify the set of synergies that affect to our example into Large, Medium and Small synergies, no matter if they are positive or negative. Then we try to apply as many Large synergies as possible. A synergy is not applied if it involves a variable for which a synergy has already been applied for that class. Once this has been done we repeat the same process for medium and finally for small synergies.

Algorithmically, it can be expressed as shown in Algorithm 5. The operations in the algorithm are calculated by using a representative for each interval. We tested two approaches, one taking as representative a value in the center of each interval and the other taking the value of the class that is nearer to equality. Using the discretization values of Figure 5.1, our first method will give as representative of "much less" 0.165, as representative of "somewhat less" 0.5 and so on, while for the same values, the second method will choose 0.33 as representative of "much less", 0.67 for "somewhat less"and so on. The second approach performed better empirically, consequently it is the one we propose and describe

in our empirical results. We will note influences and synergies calculated following this qualitativization process as $QInfluence$ and $QSynergy$ to differentiate them from their quantitative counterparties.

When our Qualitative Bayesian Classifier (QBC) is restricted to influences we call it First Order QBC (FOQBC), when synergies of two variables are applied we call it Second Order QBC (SOQBC). The development for order greater than second is not trivial because different developments of the greater order approximations are possible. This development remain as future work.

5.5 Empirical comparison

We have evaluated the classification accuracy for First and Second Order QBC and compared it with the naive Bayes classifier, as well as with other widely used machine learning algorithms. Our experiment consists in evaluating the average accuracy of each classifier, as well as its standard deviation for 15 datasets from the Irvine repository. Some information regarding these datasets can be seen in Table 5.1 For each dataset and classification method we performed 50 runs, keeping the 70% of the dataset as training set and the remaining 30% as test set. We included our algorithms in the $\mathcal{MLC}$++ (Kohavi et al., 1994) library, and used the facilities this library provides for machine learning experimentation. We compared the First and Second Order QBC with the well known machine learning algorithms CN2 (Clark and Niblett, 1989; Clark and Boswell, 1991), naive Bayes classifier, IBL (Aha et al., 1991), and ID3 (Quinlan, 1986).

Dataset	Attributes	Instances	Classes	Missing
BREAST-CANCER	9	286	2	none
BREAST	10	699	2	16
CRX	15	690	2	few
GLASS2	10	214	2	none
HEART	13	270	2	none
HYPOTHYROID	25	3162	2	some
IRIS	4	150	3	none
MONK1	6	432	2	none
MONK2	6	432	2	none
MONK3	6	432	2	5%
PARITY 5+5	10	100	2	none
SOYBEAN-LARGE	35	316	19	some
SOYBEAN-SMALL	35	47	4	none
VOTES	16	435	2	few
WAVEFORM-21	21	5000	3	none

Table 5.1: Domains used for the Qualitative Bayes empirical comparison

The ranking table (Table 5.2) is more illustrative of the results. In this table

we show how many times each classification algorithm ranked in the position indicated by the column identifier. In this table "1" means the algorithm ranked once in the position specified by the column under which it appears, "22" that it ranked twice , "333" that it ranked three times and so on. The rows are ordered with the best method on the top and the worst at the bottom, being this calculated by assigning a value to each position and adding those values.

	First	Second	Third	Fourth	Fifth	Sixth
ID3	333	333	55555	1	333	
IBL	4444	333	22	333	1	22
SOQBC	333	22	1	55555	333	1
CN2	22	333	333	22	1	4444
BAYES	333	22	1	22	4444	333
FOQBC		22	333	22	333	55555

Table 5.2: Ranking table of FOQBC, SOQBC and state of the art classifiers

5.5.1 Result analysis and justification

Table 5.2 shows that SOQBC achieves a reasonable accuracy level compared to the other classifiers used in the evaluation, being the best one in 3 out of 15 times, so it can be considered as a valuable alternative to these methods. As our main aim is to improve the interpretability of our method, this accuracy results are good enough. In the next section we give an example of the interpretability and explanation abilities of our approach.

On the other hand, we consider the FOQBC results very good considering the simplicity of the classifier induced. Its results are not too far from the ones given by continuous Bayes (surprisingly four times FOQBC outperforms naive Bayes). FOQBC makes a extremely inexact estimation of the probabilities, but has classification results comparable to the ones given by more complex classifiers. Our intuition is that the difference in performance between FOQBC and naive Bayes will increase when the number of examples in the dataset increases but this remains untested.

5.6 Examples of explanations and characterizations

5.6.1 Qualitative influences for characterization

One of the first steps carried by data analysts in the KDD process consists in getting familiarized with the data at hand. Qualitative influences can be used to get a first idea of the relationships that hold between the data. For example, if we draw the results of the FOQBC in a graph like the ones used by Neufeld, we

can have a first idea of the most relevant facts affecting each class. Displaying only large influences, we can have a graph for each class as can be seen in Figures 5.2, 5.3 and 5.4.

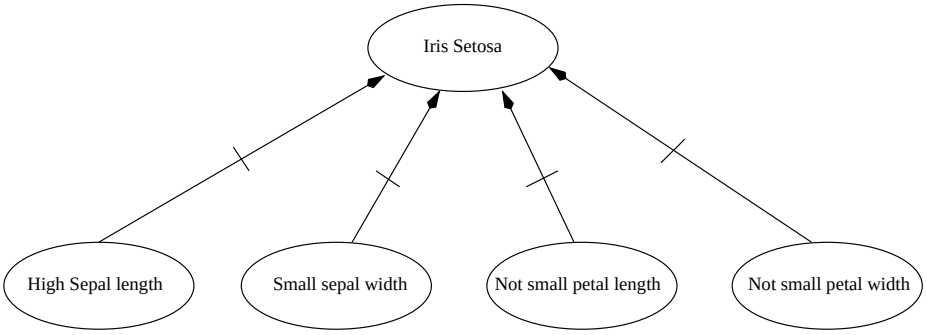


Figure 5.2: Iris setosa class

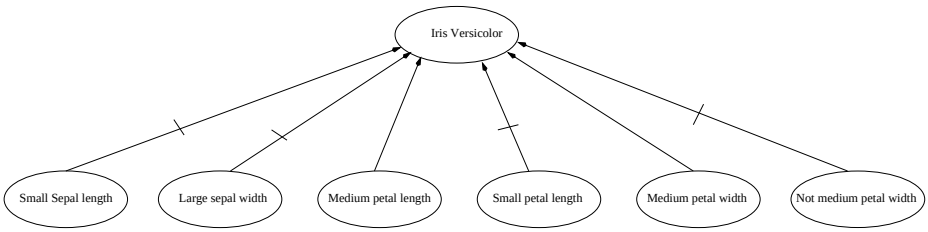


Figure 5.3: Iris versicolor class

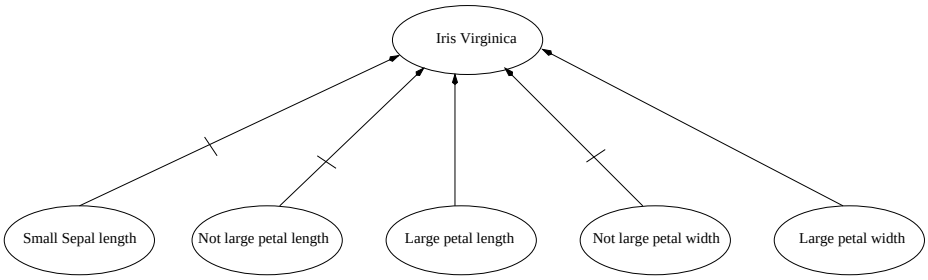


Figure 5.4: Iris Virginica class

We find those graphs easily understandable with little practice. We can also generate some other graphs isolating the more important attributes to see their interactions, or showing medium and small influences and synergies by changing the arrow width.

5.6.2 Explanation with qualitative influences and synergies

A key characteristic for a classification algorithm, for instance when it is used in decision-making, is its ability to explain why something has been classified into a class and not into another. We can use qualitative influences and synergies to explain why we classify an example into a class. For instance, suppose we have the following instance:

High sepal length, Small sepal width, High petal length, High petal width. Class: Virginica

Our system is capable to explain that it has classified this instance in the Virginica class because its high petal length and width made this class much more likely than the others. If we ask the system why it has rejected the Setosa class in this example, it answers that all the attribute values pointed that it must be rejected. Finally if we ask it why it has rejected the Versicolor class, it answers that its High petal width, as well as the coincidences (synergies) of High petal width with High sepal length and of Small sepal width with High petal length strongly discourage the class.

We have used only large influences and large synergies to show its characterization and explanation power. Of course, if the user of our system requires a more detailed characterization or explanation, medium and small influences can be used.

5.6.3 Comparison with c4.5rules results

In order to further validate the knowledge induced by FOQBC and SOQBC, we will compare our graphs in figures 5.2,5.3,5.4 with the rules extracted by c4.5rules (Quinlan, 1992). C4.5 is a well known decision tree induction algorithm and c4.5rules is a program that generates a set of pruned rules given a decision tree learned by c4.5, and then sifts this set in an attempt to find the most useful subset of them. The ruleset resulting from running c4.5rules over the iris dataset is shown in figure 5.5.

The main difference between both approaches is that influence graphs capture negative relations while c4.5rules presents only a set of positive rules. From the three rules learned by c4.5rules, rules 2 and 3 are also present in the influence graphs while rule 1 is present as a negative rule (the rightmost node of the influence graph for the Iris Setosa means that if petal width is not small then it is less likely that the instance is in the Iris Setosa class). For this example the influence graphs provide more information than the rules while the rules provide a more succint summary. A classifier based in the rules will focus mainly on petal-width disregarding any other attribute. A classifier based in the influence graphs will take into account every attribute if there is a significant influence for the attribute and the class.

```
Rule 1:
       petal-width = -Inf-0.75
       -> class Iris-setosa [96.1%]
Rule 3:
       petal-width = 1.65-Inf
       -> class Iris-virginica [95.3%]
Rule 2:
       petal-width = 0.75-1.65
       -> class Iris-versicolor [86.3%]

Default class:  Iris-setosa
```

Figure 5.5: Rules induced by C4.5rules

5.7 Summary

We have introduced qualitative influences and synergies based on the absolute orders of magnitude model. We have constructed a classifier based on this two ideas and we have shown that their accuracy results are good. We have also seen that qualitative influences and synergies can in some cases improve the comprehensibility of probabilistic reasoning. These facts make us believe that they can be useful for data mining and machine learning. We have tried to summarize in Figure 5.6 the interpretation of the results in this chapter in terms of the three dimensional space for classifiers defined in the introduction. We can see that, with respect to naive Bayes, in the case of FOQBC we are losing accuracy to achieve higher understandability and in the case of SOQBC we are increasing complexity to achieve higher understandability.

.

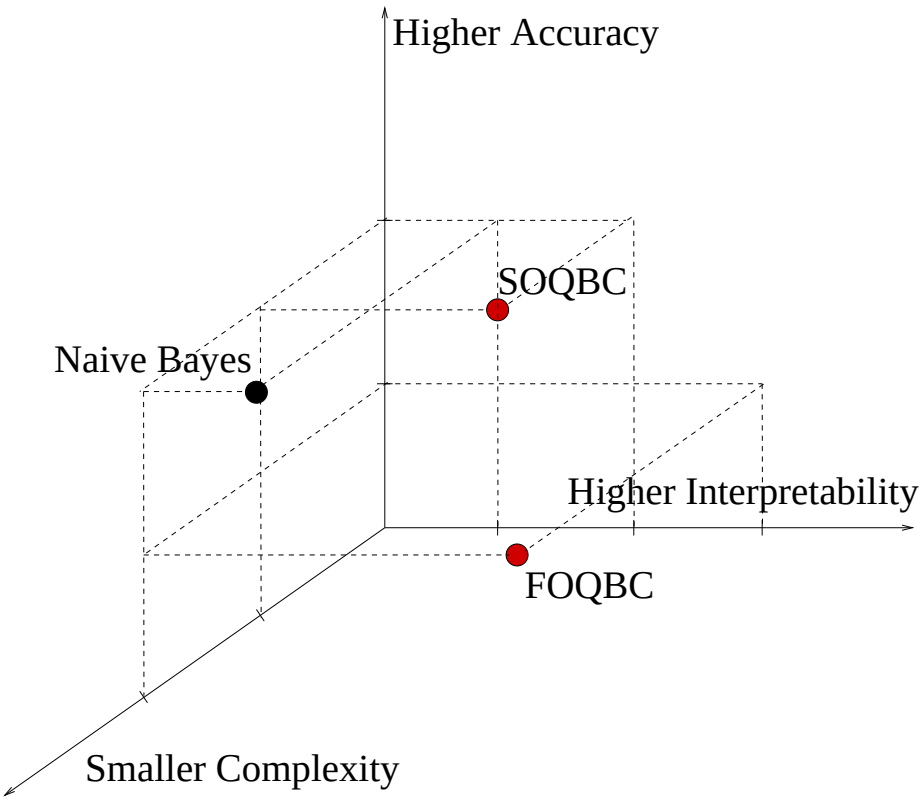


Figure 5.6: Relative positioning of the FOQBC and SOQBC with respect to naive Bayes

Chapter 6

The Indifferent Bayesian Classifier

The opposite of love is not hate, it's indifference.
The opposite of art is not ugliness, it's indifference.
The opposite of faith is not heresy, it's indifference.
And the opposite of life is not death, it's indifference.

Elie Wiesel

In this chapter we apply probability theory as extended logic, reviewed in chapter 2, to the naive Bayes classifier. The main objective of the chapter is to apply Bayesian probability theory to develop a classifier based on the assumption of conditional independence between the attributes given the class. In section 6.1 we introduce the naive Bayes model, paying special attention to its conditional independence assumptions and to the estimation of its parameters. In section 6.2 we introduce naive distributions, which is a family of probability distributions over the set of naive Bayes models, and show that they are conjugate with respect to the naive Bayes model and that they can be integrated in closed form to get averaged predictions. In section 6.3 we apply the principle of indifference getting the final expression for the Indifferent Naive Bayes classifier (INDIFFERENTNB from now on). In section 6.4 we perform the empirical comparison of INDIFFERENTNB with the standard implementation of naive Bayes and the one proposed in (Kontkanen et al., 1998) showing that INDIFFERENTNB provides a similar error rate and approximates better the probabilities, specially when little data is available. We finish with some conclusions and possibilities for future research in section 6.5.

6.1 The naive Bayes model

As has been seen in section 3.3, the naive Bayes classifier (Langley et al., 1992) is a classification method based on the assumption of conditional independence between the different variables in the dataset given the class. Following the notation in (Cowell et al., 1999), being $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ random variables we will write $\mathcal{X} \perp\!\!\!\perp \mathcal{Y}|\mathcal{Z}$ for "$\mathcal{X}$ is conditionally independent on $\mathcal{Y}$ given $\mathcal{Z}$". In this notation, the naive Bayes model states that

$$\forall i, j \;\; 1 \leq i, j \leq n \; ; \; \mathcal{A}_i \perp\!\!\!\perp \mathcal{A}_j|\mathcal{C} \tag{6.1}$$

6.1.1 The naive Bayes model as a Bayesian network

As can be seen in (Cowell et al., 1999) and in (Friedman et al., 1997) in terms of Bayesian networks, the naive Bayes model can be represented as the network in Figure 6.1. This can be easily verified using the local directed Markov property

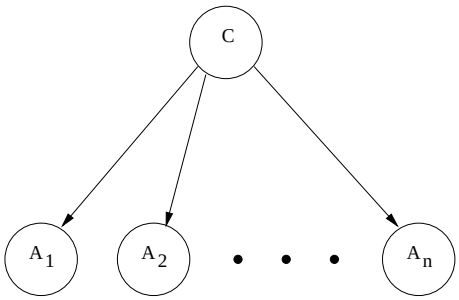


Figure 6.1: Representation of the independence assumptions under a naive Bayes model as a Bayesian network

(Cowell et al., 1999) that states that a Bayesian network encodes independence assumptions equivalent to stating that any variable in the network is independent of its non-descendants given its parents. Using this same property, the Bayesian network in Figure 6.1 is not the only Bayesian network that encodes the conditional independence assumptions in equation 6.1. In fact any one of the networks in Figure 6.2 does also satisfy the assumptions in equation 6.1.

6.1.2 The naive Bayes model as a Markov network

The conditional independence assumptions in equation 6.1 give no causal information which can be used to prefer any of the different Bayesian networks that encode them. If instead of representing these conditional independence assumptions as a Bayesian network we choose to represent them as a Markov network, the only network encoding the assumptions in equation 6.1 can be seen in Figure 6.3. In our opinion, the use of Figure 6.1 as representation of the naive

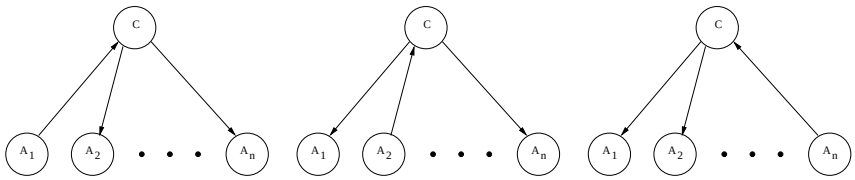


Figure 6.2: Alternative representations of the independence assumptions under a naive Bayes model as a Bayesian network

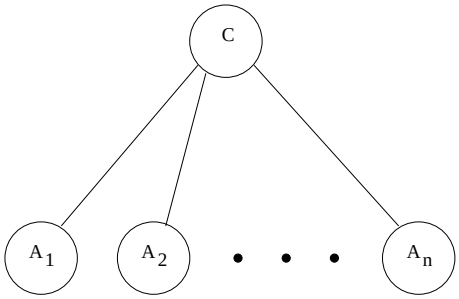


Figure 6.3: Representation of the independence assumptions under a naive Bayes model as a the Markov network

Bayes model, that is correct when interpreted in terms of acausal Bayesian networks, is slightly confusing, due to the fact that if it is interpreted in terms of causal Bayesian networks it provides more information than the conditional independence assumptions in equation 6.1. We alternatively propose to represent the naive Bayes model by the Markov network in Figure 6.3 that avoids such misunderstandings. Furthermore, the Markov network in Figure 6.3 is also the *essential graph* (in the sense of (Andersson et al., 1995)) of this equivalence class of Bayesian networks.

6.1.3 Naive Bayes parameters

Let C be the class attribute, $V = \{A_1, \ldots, A_n\}$ the set of attributes and $\mathcal{C}$ and $\mathcal{A}_i$ random variables over C and A_i respectively. Under the multinomial assumption (see (Heckerman et al., 1995)), a naive Bayes model M can be characterized by the following assumptions, parameters and constraints:

- The conditional independence assumptions in Equation 6.1.
- For each class $c \in C$:
  - The model has a parameter $\alpha_c = P(\mathcal{C} = c|M)$.

- For each attribute A_i, $1 \leq i \leq n$:
  - The model includes a set of parameters

$$\phi_{i,v,c} = P(\mathcal{A}_i = v \wedge \mathcal{C} = c|M) \tag{6.2}$$

    one for each possible value $v \in A_i$.
  - The model includes the constraint that $\alpha_c = \sum_{v \in A_i} \phi_{i,v,c}$.
- The model includes the constraint that $\sum_{c \in C} \alpha_c = 1$.

From now on we will use the term naive Bayes model to refer to this set of assumptions, parameters and constraints. We will note $\Phi = \{\phi_{i,v,c} | c \in C; 1 \leq i \leq n; v \in A_i\}$. It should be noted that α_c is introduced only in order to ease understanding and notation, because it can be determined given Φ by means of the constraints.

Imagine we need to know the probability of an unclassified observation S being in class s_C given a naive Bayes model M. Applying the independence assumptions and substituting the parameters we have that

$$p(\mathcal{C} = s_C, \mathcal{V} = S|M) = \alpha_{s_C} \prod_{i=1}^{n} \frac{\phi_{i,s_i,s_C}}{\alpha_{s_C}} \tag{6.3}$$

In many cases, the naive Bayes classifier has suffered from "the mind projection fallacy", to use the term introduced in (Jaynes, 1996). Hence, it has been accepted that what we need to do is to approximate α_j and $\phi_{i,v,j}$ by the frequencies in the data set. Defining $N_C(c)$ as the number of observations in class c in the dataset and $N_{i,C}(v, c)$ as the number of observations with class c and value v for attribute A_i, the maximum likelihood naive Bayes approximates α_c, $\phi_{i,v,c}$ as follows:

$$\alpha_c = \frac{N_C(c)}{\sum_{c' \in C} N_C(c)} \tag{6.4}$$

$$\phi_{i,v,c} = \frac{N_{i,C}(v, c)}{N_C(c)} \; \alpha_c \tag{6.5}$$

It has been empirically noticed that approximating these probabilities by their frequencies in the dataset can lead to a value of zero in expression (6.3). This can happen if one of the ϕ_{i,s_i,s_C} is zero, that is if the value of one of the attributes A_i of the new observation S, that we are trying to classify, has not been observed in the dataset for the class s_C. In other words, if the number of observations in the dataset fulfilling $A_i = s_i$ and $C = s_C$ is zero. To avoid this problem, a "softening" consisting in assigning a small probability instead of zero to ϕ_{i,s_i,s_C} can be done. That softening can improve the accuracy of the classifier. A set of *ad hoc* not well founded softening methods have been tried (Cestnik, 1990; Kohavi et al., 1997).

In (Kontkanen et al., 1998), Kontkanen et al. propose an approach for Instance Based Learning (IBL) and apply it to the naive Bayes classifier. This approach is based on the Bayesian model averaging principle (Hoeting et al., 1998). They accept the Bayesian network in Figure 6.1 plus an assumption equivalent to the Dirichlet assumption as appears in (Heckerman et al., 1995). More concretely, they define $\theta_{i,v,c} = \frac{\phi_{i,v,c}}{\alpha_c}$ and arrive to the conclusion that if we accept a Dirichlet prior for $\alpha_{\cdot}$ and for each $\theta_{i,\cdot,c}$, that is if $(\alpha_1, \ldots, \alpha_{\#C}) \sim \mathrm{Di}(\mu_1, \ldots, \mu_{\#C})$ and $(\theta_{i,1,c}, \ldots, \theta_{i,\#A_i,c}) \sim \mathrm{Di}(\sigma_{i,1,c}, \ldots, \sigma_{i,\#A_i,c})$ where μ_c, $\sigma_{i,v,c}$ are the prior hyperparameters, then the classifier resulting from applying Bayesian model averaging can be represented as a naive Bayes with the following softened approximation of α_c, $\phi_{i,v,c}$ (Kontkanen et al., 1998):

$$\alpha_c = \frac{N_C(c) + \mu_c}{\sum\limits_{c' \in C} (N_C(c') + \mu_{c'})} \tag{6.6}$$

$$\phi_{i,v,c} = \frac{N_{i,C}(v,c) + \sigma_{i,v,c}}{N_C(c) + \sum\limits_{v' \in A_i} \sigma_{i,v',c}} \; \alpha_c \tag{6.7}$$

Kontkanen's work sheds some light on why "softening" improves accuracy and shows that accuracy can be further improved if the "softening" has a theoretically well founded basis.

In spite of pointing in the right direction, in our opinion, Kontkanen et al. disregard the fact that the application of the Dirichlet assumption assumes a certain causal meaning in the direction of the edges in a Bayesian network. In fact, applying the same assumption to any of the Bayesian networks in Figure 6.2, which encode the same set of conditional independences, will provide a different result. In addition to that, the situation allows for the application of the principle of indifference. First enunciated by Bernoulli and afterwards advocated for by Laplace, Jaynes and many others (Jaynes, 1996), the principle of indifference, also known as the principle of insufficient reason tell us that if we are faced with a set of exhaustive, mutually exclusive alternatives and have no significant information that allow us to differentiate any one of them, we should assign all of them the same probability. As has been demonstrated in (Jaynes, 1996) and in (Bernardo, 2003), the principle of indifference can be seen as a special case of the more general objective Bayesian techniques of maximum entropy and reference analysis.

In the following section we show that accepting the naive Bayes model as defined above, it is possible to find a family of probability distributions that is conjugate to the model and that allows for a closed calculation of the Bayesian model averaging. After that we see that the principle of indifference suggests that the prior to be used is in this family of distributions. We will see that under this setting an additional relationship between the hyperparameters appears that has not been noticed in (Kontkanen et al., 1998).

6.2 Naive distributions

In this section we introduce naive distributions which are a family of probability distributions over the set of naive Bayes models and show that they have two key characteristics:

- They allow for the tractable averaging of naive Bayes models in order to compute the probability of an unseen example.
- They are conjugate to the naive Bayes model, hence allowing to be learned from data.

A naive distribution over a classified discrete domain Ω_C is defined by a hyperparameter set $\mathbf{N}' = \{N'_{i,C}(v,c)|1 \leq i \leq n; v \in A_i; c \in C\}$ that fulfills the following condition: Defining $N'_C(c)$ as

$$N'_C(c) = \sum_{v \in A_0} N'_{0,C}(v,c) \tag{6.8}$$

$\mathbf{N}'$ should fulfill

$$\forall i \; N'_C(c) = \sum_{v \in A_i} N'_{i,C}(v,c) \tag{6.9}$$

We will say that $P(M|\xi)$ follows a naive distribution with hyperparameter set $\mathbf{N}'$ if and only if the probability for a concrete naive Bayes model comes given by

$$P(M|\xi) = \mathcal{K} \prod_{c \in C} {\alpha_c}^{N'_C(c)} \prod_{i=1}^{n} \prod_{v \in A_i} \left(\frac{\phi_{v,i,c}}{\alpha_c} \right)^{N'_{i,C}(v,c)} \tag{6.10}$$

where $\mathcal{K}$ is a normalization constant defined by:

$$\mathcal{K} = \frac{\Gamma\left(N' + \#C \cdot [\sum_{i=1}^{n} (\#A_i) - n + 1] + 1\right)}{\prod_{c \in C} \Gamma\left(N'_C(c) + \sum_{i=1}^{n} (\#A_i) - n + 1\right)} \prod_{c \in C} \prod_{i=1}^{n} \frac{\Gamma\left(N'_C(c) + \#A_i\right)}{\prod_{v \in A_i} \Gamma\left(N'_{i,C}(v,c) + 1\right)} \tag{6.11}$$

We should remember that the parameterization of the naive Bayes classifier depends only on $\Phi = \{\phi_{i,v,c}|1 \leq i \leq n; v \in A_i; c \in C\}$ because $\forall c \; \forall i \; \alpha_c = \sum_{u \in A_i} \phi_{i,u,c}$. Naive distributions are a hyper Markov law in the sense of (Dawid and Lauritzen, 1993), for the Markov network in Figure 6.3.

6.2.1 Calculating probabilities with naive distributions

Assume that our data is generated by a naive Bayes model and that $P(M|\xi)$ follows a naive distribution with hyperparameter set $\mathbf{N}'$. We can calculate the

probability of an observation S, s_C given ξ by averaging over the set of naive Bayes models

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\xi) = \int_{M \in \mathcal{M}} P(\mathcal{V} = S, \mathcal{C} = s_C|M)P(M|\xi) \tag{6.12}$$

Solving the integral we have that

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\xi) = \mathcal{K}' \left(N'_C(s_C) + 1 + \sum_{i=1}^{n} \#A_i - n \right) \prod_{i=1}^{n} \frac{N'_{i,C}(s_i, s_C) + 1}{N'_C(s_C) + \#A_i} \tag{6.13}$$

and that

$$P(\mathcal{C} = s_C|\mathcal{V} = S, \xi) = \mathcal{K}'' \left(N'_C(s_C) + 1 + \sum_{i=1}^{n} \#A_i - n \right) \prod_{i=1}^{n} \frac{N'_{i,C}(s_i, s_C) + 1}{N'_C(s_C) + \#A_i} \tag{6.14}$$

where $\mathcal{K}'$ and $\mathcal{K}''$ are normalization constants given by:

$$\mathcal{K}'' = \frac{1}{\sum\limits_{c \in C} \left[\left(N'_C(c) + 1 + \sum\limits_{i=1}^{n} \#A_i - n \right) \prod\limits_{i=1}^{n} \frac{N'_{i,C}(s_i, c)+1}{N'_C(c)+\#A_i} \right]} \tag{6.15}$$

$$\mathcal{K}' = \mathcal{K}'' P(\mathcal{V} = S|\xi) \tag{6.16}$$

The development for this result can be found in section A.2

6.2.2 Learning with naive distributions

Given that our data is generated by a naive Bayes model, that $P(M|\xi)$ follows a naive distribution with hyperparameter set $\mathbf{N}'$ and that $\mathcal{D}$ is a dataset containing independent identically distributed complete observations over a classified discrete domain Ω_C, the posterior probability over models given $\mathcal{D}$ and ξ ,$P(M|\mathcal{D}, \xi)$, follows a naive distribution with hyperparameter set $\mathbf{N}'^*$ where:

$$N'^*_{i,C}(v, c) = N_{i,C}(v, c) + N'_{i,C}(v, c) \tag{6.17}$$

The development for this result can be found in section A.3.

6.3 The Indifferent Naive Bayes Classifier

In the case of naive Bayes models, the principle of indifference tells us that, in the lack of better information, we should assign an equal probability to every naive Bayes model, that is

$$\forall M \in \mathcal{M} \quad p(M|\xi) = Q \tag{6.18}$$

where Q is a constant. Analyzing equation 6.10, we can see that a naive distribution having

$$\mathbf{N}' = \{N'_{i,C}(v,c) = 0 | 1 \leq i \leq n; v \in A_i; c \in C\} \tag{6.19}$$

assigns an equal probability to every naive Bayes model.

INDIFFERENTNB is defined by accepting the prior probability distribution over the set of models to follow a naive distribution with parameter set $\mathbf{N}'$ given by 6.19, using the result in section 6.2.2 to calculate the posterior and using the result in section 6.2.1 to predict once the posterior is calculated.

It is easy to see that the classifier can be represented by a naive Bayes model that uses the following softened approximations:

$$\alpha_c = \frac{N_C(c) + 1 + \sum_{i=1}^{n} \#A_i - n}{\sum_{c' \in C} (N_C(c') + 1 + \sum_{i=1}^{n} \#A_i - n)} \tag{6.20}$$

$$\phi_{i,v,c} = \frac{N_{i,C}(v,c) + 1}{N_C(c) + \#A_i} \; \alpha_c \tag{6.21}$$

Comparing these results with the ones from (Kontkanen et al., 1998) shown in equations 6.6,6.7 it is worth noticing the following two facts:

- In (Kontkanen et al., 1998), Kontkanen et al. assume a Dirichlet prior distribution with a set of hyperparameters that have to be fixed at some point in time. This means that a methodological usage of that classifier requires an assessment of the prior hyperparameters for each dataset in which we would like to apply it. Instead, we have used the principle of indifference to obtain a prior without information about the dataset besides the number of attributes and the cardinality of its attributes and class.

- In equations 6.6 and 6.7 the hyperparameters $\mu_{\cdot}$ and $\sigma_{i,\cdot,c}$, for $\alpha_{\cdot}$ and $\theta_{i,\cdot,c}$ are not related. Instead, in our approach there is a link between the softening parameters, because the value of α_c in equation 6.20, depends not only on the number of classes but also on the number of attributes, n, and on the number of values of each attributes, $\#A_i$, and the value of $\theta_{i,v,c}$ in equation 6.21, depends also on the number of values of the attribute, $\#A_i$.

Furthermore, assuming a naive distribution is compatible with any of the different Bayesian networks encoding the independence assumptions in a naive Bayes model and provides the same result for all of them, because no additional causal information is assumed from the direction of the edges in the network. The experimental results in the next section show that these facts lead to a better approximation of the probabilities of the different classes when classifying.

6.4 Experimental results

We tested three algorithms over 17 datasets from the Irvine repository (Blake et al., 1998). To discretize continuous attributes we used equal frequency discretization with 5 intervals. For each dataset and algorithm we tested both error rate and *LogScore*. *LogScore* is calculated by adding the minus logarithm of the probability assigned by the classifier to the correct class and gives an idea of how well the classifier is estimating probabilities (the smaller the score the better the result). If we name our test set $\mathcal{D}'$ we have

$$LogScore(M, \mathcal{D}') = \sum_{(S,s_C)\in\mathcal{D}'} -\log(P(\mathcal{C} = s_C|\mathcal{V} = S, M)) \tag{6.22}$$

We used 10 fold cross validation, and for each fold we ran the algorithms with the complete learning fold, with a random sample of 50% of the data in the learning fold and with a random sample of 10% of the data in the learning fold and then measured error rate and *LogScore* against the test fold. This is done because the three methods converge to the same model given enough data. Hence, comparing them when the size of the training data is small can provide us with good insight on how they differentiate.

The error rates appear in tables 6.2, 6.4 and 6.6, with the best method for each dataset boldfaced. *LogScore*'s appear in tables 6.3, 6.5 and 6.7. The columns of the tables are the induction methods and the rows are the datasets. The meaning of the column headers are:

- MLNB is the standard naive Bayes algorithm using frequencies as probability estimates, as shown in equations 6.4 and 6.5.
- BIBL is the algorithm appearing in (Kontkanen et al., 1998) and shown in equations 6.6 and 6.7 and fixing the hyperparameters to get uniform prior probability distributions.
- INDIFFERENTNB is the Indifferent Naive Bayes as described in equations 6.20 and 6.21.

6.4.1 Dataset description

In order to ease the interpretation of the results, we provide a short presentation of each of the datasets used for the comparison. They were randomly selected from the UCI repository. In table 6.1 we provide a short summary of its main characteristics. The datasets used were:

adult This data was extracted from the census bureau database. Prediction task is to determine whether a person makes over 50K a year. 48842 instances, mix of continuous and discrete attributes.

australian Australian Credit Approval. This data is very similar to the crx dataset but:

- missing values have been replaced with the medians, which is unfair to the algorithms that can deal with missing data well. Replacing an attribute by its mean/median value is known to be one of the poorest methods of handling missing values.
- attribute 4 is removed (in the entire dataset atts 4 and 5 were completely correlated)
- discrete attribute values are numbered in increasing likelihood of being class +.

690 instances, mix of continuous and discrete attributes.

breast Wisconsin Breast Cancer Database (January 8, 1991). Each instance has one of 2 possible classes: benign or malignant. 699 instances, all the attributes are continuous.

car Marko Bohanek car dataset. Prediction task is to determine a car acceptability in an ordinal scale (unacc, acc, good, v-good).1728 instances, all attributes are discrete.

chess Chess End-Game – King+Rook versus King+Pawn on a7. The pawn on a7 means it is one square away from queening. It is the King+Rook's side (white) to move. Prediction task is to determine whether White-can-win ("won") and White-cannot-win ("nowin"). It is assumed that White is deemed to be unable to win if the Black pawn can safely advance. 3196 instances, all attributes discrete.

cleve Dr. Detrano's database modified by Brian Frasca (6/13/94) - Last attribute deleted to give a two-class problem as suggested by Holte. Prediction task is to determine whether the patient is healthy or sick. 303 instances, mix of continuous and discrete attributes.

crx This dataset concerns credit card applications. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. 690 instances, mix of continuous and discrete attributes.

flare Solar flare database (UCI flare.data1). 323 instances, all the attributes are discrete.

glass Glass Identification Database. The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence if it is correctly identified. 214 instances, mix of continuous and discrete attributes.

glass2 Glass Identification Database. Removed first attribute, combined classes 1 and 3, and removed classes 4 through 7 as suggested by Holte. This turns the problem into predicting whether the window was float processed or not. Because classes 4-7 were deleted, there are less instances than in glass. 163 instances, all attributes continuous.

iris Iris Plants Database. Predicted attribute: class of iris plant. 150 instances, all attributes are continuous.

letter Letter Image Recognition Data. The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. 20000 instances, all attributes are discrete with values 0..15.

liver BUPA liver disorders. The first 5 variables are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. Each instance constitutes the record of a single male individual. 345 instances, all attributes continuous.

nursery Nursery database was derived from a hierarchical decision model originally developed to rank applications for nursery schools. It was used during several years in 1980's when there was excessive enrollment to these schools in Ljubljana, Slovenia, and the rejected applications frequently needed an objective explanation. The final decision depended on three subproblems: occupation of parents and child's nursery, family structure and financial standing, and social and health picture of the family. 12960 instances, all attributes discrete.

primary-tumor Primary Tumor Domain. This primary tumor domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data. 339 instances, all attributes discrete.

soybean Large Soybean Database. 683 instances, all attributes discrete.

votes Voting records drawn from the Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, Volume XL: Congressional Quarterly Inc. Washington, D.C., 1985. This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The prediction task is determining whether the voter is democrat or republican. 435 instances, all attributes discrete.

6.4.2 Interpretation of the results

The conclusions that are drawn from the statistical significance analysis of all the numbers in tables 6.2, 6.3, 6.4, 6.5, 6.6 and 6.7 are the following:

- Error rate results are similar for the three algorithms, being the only relevant difference that BIBL seems to improve MLNB significantly when using 10% of the learning data whilst MLNB improves slightly BIBL with 100% of the learning data.
- Both BIBL and INDIFFERENTNB have a better *LogScore* than MLNB.

Dataset	Attributes	Instances	Classes	Missing
ADULT	14	48842	2	some
AUSTRALIAN	15	690	2	none
BREAST	10	699	2	16
CAR	6	1728	4	none
CHESS	36	3196	2	none
CLEVE	13	303	2	some
CRX	15	690	2	few
FLARE	10	1063	8	none
GLASS	10	214	7	none
GLASS2	9	163	2	none
IRIS	4	150	3	none
LETTER	16	20000	26	none
LIVER	7	345	2	none
NURSERY	8	12960	5	no
PRIMARY-TUMOR	18	339	22	some
SOYBEAN	35	683	19	some
VOTES	16	435	2	few

Table 6.1: Datasets information

- INDIFFERENTNB has a better *LogScore* than both BIBL when using less than 100% of the learning data.

In the following we illustrate each point separately.

Error rate comparison

We can see the results of the statistical significance t-test at 5% in table 6.8. The columns are the amount of training data used and the rows the algorithms compared. If we are looking at the row that contains the comparison INDIFFERENTNB vs. BIBL, the entry at column 10% means that, when using 10% of the training data, INDIFFERENTNB provided 4 times a result better than BIBL in a statistically significant way whilst BIBL provided 5 times a result better than INDIFFERENTNB in a statistically significant way.

Looking a the first row, BIBL has slightly better results than INDIFFERENTNB when 100% of the training data is not used. The same can be said, in this case favoring INDIFFERENTNB when comparing INDIFFERENTNB with MLNB. Comparing BIBL with MLNB, we see that BIBL improves error rate in a relevant number of datasets when 10% of the learning data is used, and that using 100% of the learning data MLNB slightly improves BIBL.

LogScore comparison

In table 6.9 we can see the statistical significance results for *LogScore*. From there we can easily conclude that both INDIFFERENTNB and BIBL improve significantly MLNB *LogScore* in most of the cases. It can also be seen that INDIFFERENTNB improves significatly BIBL *LogScore* with 10% of the learning data, that this improvement is not so relevant when using 50% of the learning data and that with 100% both classifiers improve the other in the same number of datasets. The dominance of INDIFFERENTNB over BIBL and MLNB in *LogScore* can be seen in figures 6.4 and 6.5. It is worth noticing that whilst INDIFFERENTNB consistently and significantly improves *LogScore*, this improvement, in general, does not translate into a clear improvement of the error rate. This can

Dataset	BIBL	IndifferentNB	MLNB
ADULT	**18.66 ± 0.75**	18.92 ± 0.73	18.76 ± 0.73
AUSTRALIAN	17.25 ± 0.88	**15.97 ± 0.76**	20.84 ± 0.86
BREAST	**3.00 ± 0.45**	3.15 ± 0.52	3.43 ± 0.52
CAR	20.05 ± 0.49	20.35 ± 0.67	**19.75 ± 0.55**
CHESS	15.54 ± 0.78	15.58 ± 0.78	**15.44 ± 0.79**
CLEVE	22.18 ± 0.83	**20.78 ± 1.00**	27.35 ± 1.04
CRX	17.01 ± 1.03	**15.59 ± 0.85**	21.08 ± 1.04
FLARE	**23.77 ± 1.15**	26.21 ± 1.16	24.72 ± 1.12
GLASS	41.69 ± 2.01	**40.99 ± 2.01**	73.23 ± 2.65
GLASS2	**34.68 ± 1.64**	34.74 ± 1.56	35.78 ± 1.58
IRIS	**16.13 ± 1.55**	16.27 ± 1.59	**16.13 ± 1.96**
LETTER	**31.18 ± 1.27**	33.36 ± 1.23	32.07 ± 1.24
LIVER	**44.07 ± 1.69**	44.13 ± 1.73	44.26 ± 1.67
NURSERY	10.56 ± 1.09	**10.36 ± 1.10**	10.52 ± 1.09
PRIMARY-TUMOR	**68.14 ± 1.74**	69.66 ± 1.56	79.75 ± 0.94
SOYBEAN	**28.78 ± 1.22**	36.19 ± 1.05	61.60 ± 1.59
VOTES	10.66 ± 0.81	10.46 ± 0.70	**10.06 ± 1.12**

Table 6.2: Averages and standard deviations of error rate using 10% of the learning data

Dataset	BIBL	IndifferentNB	MLNB
ADULT	686.00 ± 4.50	**678.89 ± 4.70**	778.00 ± 5.56
AUSTRALIAN	17.91 ± 0.80	**14.21 ± 0.64**	67.36 ± 1.88
BREAST	8.46 ± 0.86	**4.85 ± 0.43**	35.55 ± 2.21
CAR	**32.90 ± 0.93**	37.95 ± 0.80	65.95 ± 2.54
CHESS	**47.23 ± 1.02**	47.67 ± 1.01	52.90 ± 1.73
CLEVE	10.42 ± 0.90	**7.49 ± 0.72**	61.53 ± 2.45
CRX	18.07 ± 0.83	**14.65 ± 0.66**	65.84 ± 2.15
FLARE	47.17 ± 1.18	**44.15 ± 1.18**	176.33 ± 3.43
GLASS	17.44 ± 1.31	**10.84 ± 0.99**	247.44 ± 6.97
GLASS2	9.04 ± 0.88	**5.63 ± 0.49**	63.53 ± 3.05
IRIS	3.09 ± 0.84	**2.65 ± 0.58**	18.31 ± 2.77
LETTER	1505.51 ± 9.10	**1258.11 ± 8.10**	3727.18 ± 14.95
LIVER	16.43 ± 1.04	**12.57 ± 0.68**	64.21 ± 3.03
NURSERY	**154.84 ± 2.75**	167.21 ± 2.76	157.51 ± 3.16
PRIMARY-TUMOR	56.34 ± 2.11	**41.59 ± 1.48**	352.69 ± 4.48
SOYBEAN	87.18 ± 1.96	**81.72 ± 1.91**	812.12 ± 8.39
VOTES	13.54 ± 1.06	**12.23 ± 1.08**	40.28 ± 2.38

Table 6.3: Averages and standard deviations of *LogScore* using 10% of the learning data

Dataset	BIBL	IndifferentNB	MLNB
ADULT	**18.47 ± 0.76**	18.55 ± 0.79	18.50 ± 0.77
AUSTRALIAN	14.81 ± 0.52	**14.20 ± 0.49**	15.13 ± 0.57
BREAST	**2.55 ± 0.37**	2.66 ± 0.26	3.20 ± 0.52
CAR	15.53 ± 0.97	**14.73 ± 1.03**	15.43 ± 0.96
CHESS	13.01 ± 0.63	13.05 ± 0.64	**12.94 ± 0.60**
CLEVE	17.90 ± 1.15	**17.50 ± 1.25**	18.82 ± 1.22
CRX	14.57 ± 0.68	**14.28 ± 0.57**	15.42 ± 0.84
FLARE	**23.85 ± 0.56**	25.17 ± 0.72	24.64 ± 0.53
GLASS	21.26 ± 1.56	22.61 ± 1.89	**20.84 ± 1.31**
GLASS2	25.96 ± 1.76	**25.12 ± 1.60**	26.32 ± 1.77
IRIS	11.60 ± 0.95	12.27 ± 0.73	**10.53 ± 1.17**
LETTER	25.74 ± 1.00	27.31 ± 1.05	**25.41 ± 0.97**
LIVER	**37.92 ± 1.08**	38.32 ± 1.16	37.98 ± 1.13
NURSERY	9.75 ± 0.99	**9.68 ± 0.99**	9.75 ± 0.99
PRIMARY-TUMOR	**55.67 ± 1.52**	57.09 ± 1.71	58.63 ± 1.44
SOYBEAN	10.65 ± 0.33	12.84 ± 0.49	**7.84 ± 1.06**
VOTES	9.93 ± 0.60	10.11 ± 0.58	**9.35 ± 0.65**

Table 6.4: Averages and standard deviations of error rate using 50% of the learning data

Dataset	BIBL	IndifferentNB	MLNB
ADULT	665.65 ± 4.17	**665.22 ± 4.16**	672.95 ± 4.41
AUSTRALIAN	13.89 ± 0.45	**13.30 ± 0.38**	19.23 ± 1.28
BREAST	9.89 ± 0.57	**7.17 ± 0.41**	25.58 ± 1.45
CAR	25.23 ± 0.63	26.10 ± 0.61	**24.56 ± 0.63**
CHESS	**41.62 ± 0.47**	41.84 ± 0.46	42.33 ± 0.58
CLEVE	6.84 ± 0.68	**6.51 ± 0.63**	10.08 ± 1.17
CRX	14.20 ± 0.28	**13.25 ± 0.29**	23.81 ± 1.67
FLARE	**41.94 ± 0.86**	42.39 ± 0.74	79.69 ± 2.18
GLASS	6.53 ± 0.70	**5.86 ± 0.58**	34.10 ± 2.09
GLASS2	5.13 ± 0.65	**4.68 ± 0.58**	6.23 ± 1.04
IRIS	**1.57 ± 0.42**	1.62 ± 0.38	2.55 ± 0.78
LETTER	1046.21 ± 7.69	**1042.07 ± 7.07**	1281.05 ± 10.57
LIVER	9.96 ± 0.44	**9.81 ± 0.43**	10.72 ± 0.41
NURSERY	**148.02 ± 2.86**	150.24 ± 2.87	148.20 ± 2.83
PRIMARY-TUMOR	35.05 ± 0.90	**31.56 ± 0.89**	145.12 ± 3.76
SOYBEAN	**23.17 ± 0.96**	25.62 ± 0.77	24.37 ± 2.05
VOTES	12.30 ± 0.31	**12.15 ± 0.38**	16.79 ± 1.36

Table 6.5: Averages and standard deviations of *LogScore* using 50% of the learning data

Dataset	BIBL	IndifferentNB	MLNB
ADULT	**18.48 ± 0.71**	18.52 ± 0.73	18.50 ± 0.71
AUSTRALIAN	14.67 ± 0.56	14.52 ± 0.57	**14.49 ± 0.52**
BREAST	2.69 ± 0.33	**2.63 ± 0.26**	2.80 ± 0.27
CAR	14.51 ± 0.37	**13.69 ± 0.49**	14.35 ± 0.44
CHESS	12.16 ± 0.34	12.16 ± 0.33	**12.15 ± 0.33**
CLEVE	16.73 ± 0.87	**16.33 ± 0.70**	17.58 ± 0.76
CRX	14.46 ± 0.39	**14.25 ± 0.33**	14.72 ± 0.47
FLARE	24.54 ± 0.38	24.74 ± 0.33	**24.50 ± 0.42**
GLASS	17.33 ± 1.14	20.14 ± 1.38	**15.88 ± 0.78**
GLASS2	23.82 ± 1.21	**23.70 ± 1.12**	23.82 ± 1.21
IRIS	**11.33 ± 1.09**	11.60 ± 1.04	**11.33 ± 1.06**
LETTER	25.18 ± 0.95	26.27 ± 1.03	**24.94 ± 0.92**
LIVER	**37.99 ± 0.90**	38.04 ± 0.87	**37.99 ± 0.90**
NURSERY	9.80 ± 0.99	**9.75 ± 1.00**	9.78 ± 0.99
PRIMARY-TUMOR	**51.71 ± 1.16**	53.12 ± 0.98	54.91 ± 1.09
SOYBEAN	9.19 ± 0.47	9.89 ± 0.49	**5.22 ± 0.49**
VOTES	9.83 ± 0.44	9.93 ± 0.33	**9.65 ± 0.46**

Table 6.6: Averages and standard deviations of error rate using 100% of the learning data

Dataset	BIBL	IndifferentNB	MLNB
ADULT	666.72 ± 4.13	**666.61 ± 4.12**	669.36 ± 4.22
AUSTRALIAN	12.95 ± 0.35	**12.83 ± 0.33**	13.34 ± 0.70
BREAST	9.54 ± 0.26	**7.96 ± 0.23**	16.41 ± 0.27
CAR	24.33 ± 0.30	24.68 ± 0.29	**23.98 ± 0.30**
CHESS	**40.61 ± 0.29**	40.72 ± 0.27	40.73 ± 0.77
CLEVE	6.48 ± 0.28	**6.32 ± 0.28**	8.47 ± 0.66
CRX	13.09 ± 0.24	**12.84 ± 0.23**	15.26 ± 0.93
FLARE	**40.61 ± 0.65**	41.61 ± 0.52	61.61 ± 1.70
GLASS	**4.18 ± 0.56**	4.50 ± 0.53	11.85 ± 1.08
GLASS2	4.14 ± 0.38	**4.02 ± 0.36**	4.18 ± 0.39
IRIS	1.69 ± 0.38	**1.62 ± 0.30**	3.62 ± 0.85
LETTER	**987.71 ± 7.18**	999.45 ± 7.13	1080.25 ± 8.16
LIVER	9.30 ± 0.34	**9.28 ± 0.33**	9.31 ± 0.35
NURSERY	**147.14 ± 2.83**	148.24 ± 2.84	147.85 ± 2.77
PRIMARY-TUMOR	30.11 ± 0.56	**28.55 ± 0.35**	83.66 ± 1.08
SOYBEAN	21.17 ± 0.60	24.39 ± 0.56	**9.96 ± 1.01**
VOTES	12.06 ± 0.52	**11.98 ± 0.47**	12.38 ± 0.88

Table 6.7: Averages and standard deviations of *LogScore* using 100% of the learning data

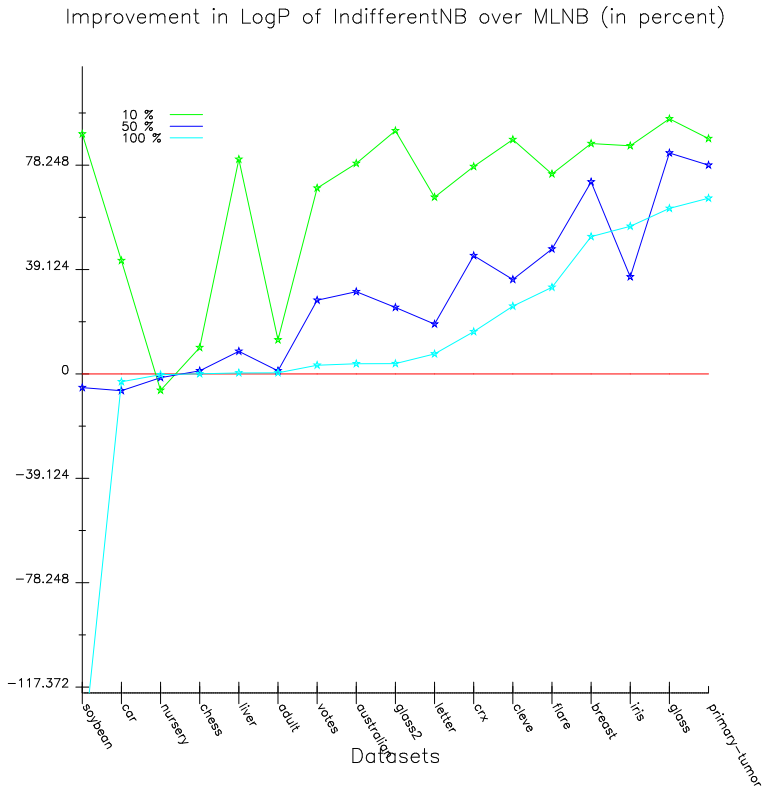


Figure 6.4: Comparison of INDIFFERENTNB and MLNB *LogScore*

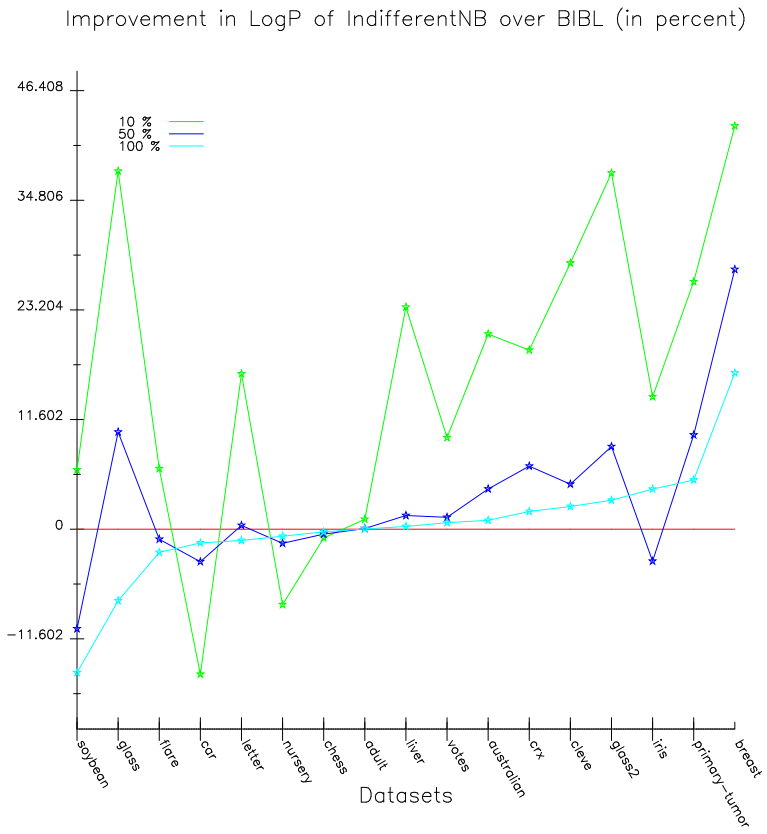


Figure 6.5: Comparison of INDIFFERENTNB and BIBL *LogScore*

be seen as an example of the already known fact (Devroye et al., 1996) that a good classifier does not need to correctly approximate the class probabilities and that classification is easier than regression function estimation. In spite of that, when using a learning algorithm, many times the user needs a good estimate of the class probabilities (the setting is closer to density estimation than to classification). This is the case, for example, in one of the most successful practical uses of learning algorithms, that is predicting the response to a marketing campaign.

Classifiers compared	10 %	50%	100%
INDIFFERENTNB VS. BIBL	4 - 5	5 - 7	5 - 5
INDIFFERENTNB VS. MLNB	7 - 4	8 - 6	5 - 5
BIBL VS. MLNB	10-2	7 - 5	2 - 6

Table 6.8: Error rate statistical significance results at 5%

Classifiers compared	10 %	50%	100%
INDIFFERENTNB VS. BIBL	14 - 3	9 - 4	7 - 7
INDIFFERENTNB VS. MLNB	16 - 1	13 - 2	11 - 2
BIBL VS. MLNB	16-0	14 - 1	12 - 2

Table 6.9: *LogScore* statistical significance results at 5%

6.5 Summary

We have developed INDIFFERENTNB, the Indifferent Naive Bayes classifier, by accurately defining the naive Bayes model based on its conditional independence assumptions and calculating a conjugate distribution for the set of models. We have used the principle of indifference to define the prior distribution. While the objective of the development was mainly theoretical we have seen that the development leads to improvements in the quality of the probabilities assigned to the class, specially when only small amounts of data are available. An interesting possibility for the future is providing INDIFFERENTNB with the possibility of handling unknown values.

Recently, Dash and Cooper have proposed a different method for averaging naive Bayes classifiers (Dash and Cooper, 2002). Empirically comparing this method with INDIFFERENTNB remains as future work.

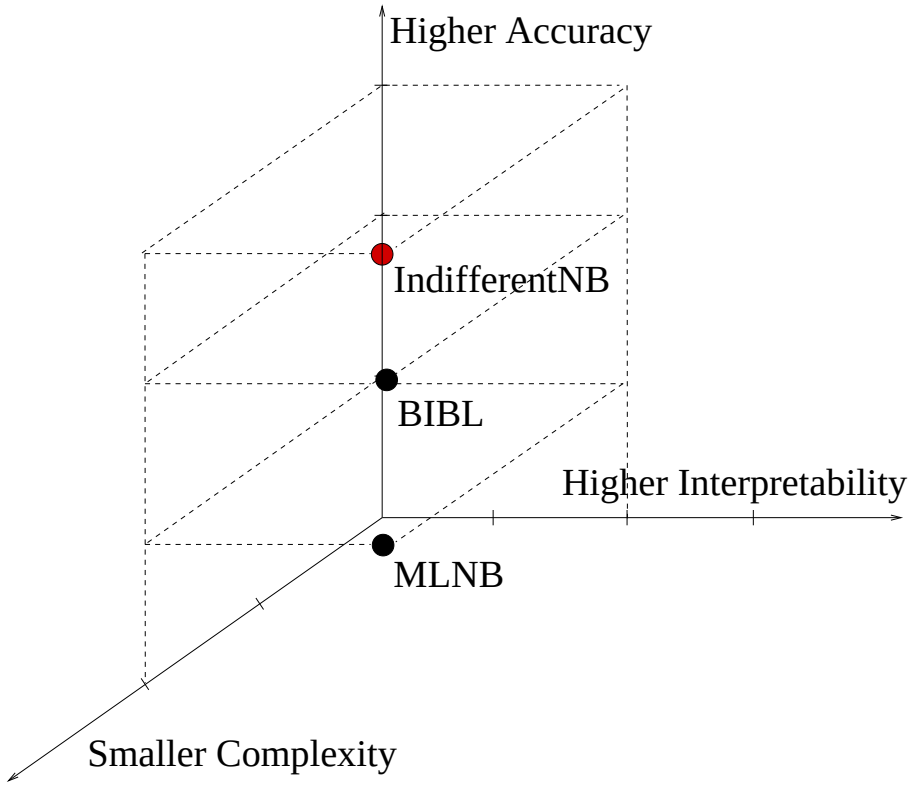


Figure 6.6: Relative positioning of the INDIFFERENTNB with respect to MLNB and BIBL

We have tried to summarize in Figure 6.6 the interpretation of the results in this chapter in terms of the three dimensional space for classifiers defined in the introduction. There we can see that BIBL improves naive Bayes accuracy and that INDIFFERENTNB improves accuracy over both naive Bayes and BIBL. In this case, these improvements come at no cost neither in complexity nor in understandability.

Chapter 7

Empirical Local Bayesian model averaging of TAN classifiers

A poet's hope: to be,
like some valley cheese,
local, but prized elsewhere.

W. H. Auden (1907 - 1973), Collected Poems

In the previous chapter we have seen that the independence assumptions under the naive Bayes model can be clearly stated, that we can account for model uncertainty under the assumption that a naive Bayes model is generating the data and the benefits that these two facts provide. In the next three chapters we present our effort to improve *Tree Augmented Naive Bayes* (TAN). Already introduced in chapter 3, TAN has shown to be competitive with state-of-the-art machine learning algorithms (Friedman et al., 1997). However, the TAN induction algorithm that appears in (Friedman et al., 1997) (and presented as well in section 3.4.1) can be improved in two ways.

The first way in which TAN can be improved is by noticing that the algorithm in section 3.4.1 does not account for model uncertainty and providing a solution for this problem. As was presented in section 3.5, from the point of view of probability theory, the way to deal with model uncertainty is through Bayesian model averaging. We have developed three alternative solutions for applying Bayesian model averaging to TAN models:

- In this chapter we will show that we can account for model uncertainty by selecting a small set of TAN structures in a neighborhood of the maximum likelihood TAN structure and empirically evaluating the probability of each structure from the sample. This results in more accurate predictions than the ones coming from the maximum likelihood TAN structure alone at the cost of increasing both the learning time and the classification time complexity.
- In chapter 8 we will see that for a concrete family of distributions over the set of models, named decomposable distributions over TANs, it is possible to calculate the result of the Bayesian model averaging in polynomial time. The development in chapter 8 is parallel to the one already presented in chapter 6 for naive Bayes. The result provided in chapter 8 provides the optimal way (from a Bayesian perspective) to account for model uncertainty when learning TAN. This comes at the cost of increasing considerably the classification time complexity.
- Finally, in chapter 9 we will see that decomposable distributions over TANs allow us to refine the algorithm presented in this chapter because they have the property of making efficiently computable the k TAN structures with maximum a posteriori probability, providing a considerable improvement in the learning time complexity associated with the algorithm, a slight improvement in accuracy and a better founded theoretical basis.

The second way in which TAN can be improved has as starting point a fact that was noted in section 3.4.1. There, we described that the usage of equation 3.9 to fix the parameters, instead of the maximum likelihood parameters given by equation 3.8, improves the accuracy of the classifier. This fact is somewhat puzzling and in our opinion, no theoretical justification was given for it. Furthermore, if the usage of the maximum likelihood principle for fixing the parameters can be improved by softening, possibly its usage in the determination of the best TAN structure can also be improved. From a Bayesian point of view, when forced to choose a model we should select the one which better represents the results of the BMA. If the posterior distribution over models is well behaved, we can select the model with maximum posterior probability, which is usually known as maximum a posteriori or MAP. In chapter 9 we prove that under decomposable distributions it is possible to efficiently calculate the MAP structure and parameters. The algorithm for finding the MAP structure and parameters presented in chapter 9 has a reduced error rate and in our opinion is theoretically better founded than STAN.

7.1 Local Bayesian model averaging for TAN induction

In this section we provide a solution to the first of the two problems of the TAN learning algorithm as shown in (Friedman et al., 1997), namely that the algo-

rithm ignores uncertainty in model selection. *Bayesian model averaging* (BMA) (Hoeting et al., 1998) provides a coherent mechanism for accounting for uncertainty in modeling. We have already introduced Bayesian model averaging in section 3.5. In this section we introduce *local Bayesian model averaging* (LBMA), a practical way of implementing BMA. After that, we see how LBMA can be applied in the case of TAN models.

7.1.1 Local Bayesian model averaging

As seen in section 3.5, BMA is how probability theory tell us we should act when faced with the problem of choosing between a set of different non-intersecting models. More formally, assuming ξ represents the hypothesis that the model underlying the data (M) is known to be in $\mathcal{M}$ we have that:

$$P(\mathcal{C} = s_C|\mathcal{D}, \mathcal{V} = S, \xi) = \int_{M \in \mathcal{M}} P(\mathcal{C} = s_C|M, \mathcal{V} = S)P(M|\mathcal{D}, \xi) \tag{7.1}$$

Given a model M, the probability of a classified observation, $P(\mathcal{V} = S, \mathcal{C} = s_C|M)$, is usually known. In order to apply equation 7.1 we have to develop $P(M|\mathcal{D}, \xi)$ into more detail. By Bayes theorem we have that

$$P(M|\mathcal{D}, \xi) = \frac{P(\mathcal{D}|M)P(M|\xi)}{\int_{M' \in \mathcal{M}} P(\mathcal{D}|M')P(M'|\xi)} \tag{7.2}$$

Here $P(M|\xi)$ is the prior probability of M, that is, our belief before seeing the data in $\mathcal{D}$ that M is the model generating the data. $P(\mathcal{D}|M)$ is the probability that model M generates the data in $\mathcal{D}$. The application of BMA in learning algorithms presents some problems, coming from the computational cost of calculating equation 7.1 by numerical integration when no closed form for the integral is available. In order to handle this problem, we propose LBMA, an heuristic approach to approximate BMA. The idea is similar in spirit to the Occam's Window method described in (Hoeting et al., 1998; Madigan and Raftery, 1994). To apply LBMA we should have an heuristic $h(M, \mathcal{D})$ such that

$$h(M, \mathcal{D}) \approx P(M|\mathcal{D}) \tag{7.3}$$

In order to approximate the summation in equation 7.1, and given that we have $h(M, \mathcal{D})$, we can define our set of *interesting models* $\mathcal{M}'$ as:

$$\mathcal{M}' = \{M \in \mathcal{M} | h(M, \mathcal{D}) \geq \gamma\} \tag{7.4}$$

γ represents a compromise between the prediction accuracy and its computational cost. It should be big enough to make $\#\mathcal{M}' \ll \#\mathcal{M}$ (sometimes $\mathcal{M}'$ will be finite and $\mathcal{M}$ infinite, but we will use the integral sign generically in our development). It (γ) should be small enough in order for

$$P(M|\mathcal{D}) \approx P'(M|\mathcal{D}) = \frac{P(\mathcal{D}|M)P(M|\xi)}{\int_{M' \in \mathcal{M}'} P(\mathcal{D}|M')P(M'|\xi)} \tag{7.5}$$

and

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\mathcal{D}, \xi) \approx \int_{M \in \mathcal{M}'} P(\mathcal{V} = S, \mathcal{C} = s_C|M) P'(M|\mathcal{D}, \xi) \tag{7.6}$$

to be accurate approximations.

It is interesting to note that maximum likelihood prediction is a concrete case of LBMA where $h(M, \mathcal{D}) = P(\mathcal{D}|M)$ and γ is implicitly set in order for $\mathcal{M}'$ to contain only a single model. We describe LBMA algorithmically in algorithm 6, assuming $\mathcal{M}'$ is finite, to ease its understanding. The algorithm calculates a weighted set of models. Once it has calculated it, we can use it to classify by calculating equation 7.6 (where the integral is substituted by a summation) for each class and choosing the class with higher probability.

Empirical local Bayesian model averaging

In algorithm 6, we can see that the computation of the probability of the dataset given the model is done by calling a function named `CalculateProb`$(\mathcal{D}, M)$. If we have a closed form expression for this function we will code `CalculateProb` specifically for the model we are averaging. Otherwise, we can empirically calculate its value as can be seen in algorithm 7. Whenever we compute `CalculateProb` as in algorithm 7 we will say that we are using empirical local Bayesian model averaging or ELBMA.

7.1.2 Empirical local Bayesian model averaging of TAN models

In this section we demonstrate the application of ELBMA to the case of TAN induction. For this concrete case, our class of models $\mathcal{M}$ is

$$\mathcal{M} = \{\langle \overline{E}^*, \Theta_{\overline{E}^*} \rangle | \overline{E}^* \in \overline{\mathcal{E}}^*, \Theta_{\overline{E}^*} \in Parameters(\overline{E}^*)\} \tag{7.7}$$

where we remind that $\overline{\mathcal{E}}^*$ is the set of all TAN structures and $\Theta_{\overline{E}^*}$ defined as in section 3.4.

We perform a first reduction of $\mathcal{M}$ accepting the use of the results in equation 3.8 or equation 3.9 depending on whether we decide to use the maximum likelihood principle or the ad hoc adjustment proposed in (Friedman et al., 1997) to fix the network parameters. In any case, we will only average over the structures, fixing the parameters by using the corresponding equation in each case.

Our heuristic over structures will be given by the function `Construct-TAN` that can be seen in algorithm 1, just modifying the step where a maximum weight spanning tree (MWST) is induced, to generate a set containing the k MWSTs. The problem of computing the k MWSTs in order is well known and different algorihtm with different time complexities have been provided. In (Katoh et al., 1981) it is demonstrated that the problem can be solved in $\mathcal{O}((\log(\beta(n^2, n)) + k) \cdot n^2)$ time for a complete graph, where $\beta(m, n)$ is defined to

```
procedure LBMA-Main (Dataset 𝒟,Real γ,Heuristic h)
    var
       WeightedSetOfModels Result;
    begin
        Calculate ℳ′ using h and γ;
        return LBMA(𝒟,ℳ′);
    end

procedure LBMA (Dataset 𝒟,SetOfModels ℳ′)
    var
       WeightedSetOfModels Result;
       ProbabilityDistribution P′;
    begin
        foreach M ∈ ℳ′
            P′(M) = CalculateProb(𝒟, M) * P(M|ξ);
        end
        Normalize P′(M);
        Result = {(M, P′(M))|M ∈ ℳ′};
        return Result;
    end
```

Algorithm 6: Local Bayesian model averaging

```
/* Calculates P(𝒟|M) */
procedure CalculateProb (Dataset 𝒟,ProbabilisticModel M)
    var
       Real P_M;
    begin
        P_M = 1;
        foreach (S, s_C) ∈ 𝒟
            P_M = P_M * P(𝒞 = s_C|M, 𝒱 = S);
        end
        return P_M;
    end
```

Algorithm 7: Empirical local Bayesian model averaging computation of probabilities

be $\min\{i|\log^{(i)} n \leq m/n\}$ and $\log^{(i)} x$ denotes the log function iterated i times. A simpler algorithm for the same problem with a slightly higher complexity is provided in (Gabow, 1977). A slightly better time complexity under certain conditions appears in (Eppstein, 1992)

In order to calculate $P'(M|\xi)$, we set a prior over tree structures that assigns the same probability to each possible tree structure (since they can be considered of a similar complexity). We also have to know how to calculate $P(\mathcal{C} = s_C|M, \mathcal{V} = S)$. The expansion of equation 3.1 taking into account the TAN structure is

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\overline{E}^*, \Theta_{\overline{E}^*}) = \theta_C(s_C)\ \theta_{\rho_{\overline{E}}|C}(s_{\rho_{\overline{E}}}, s_C) \prod_{u,v\in\overline{E}} \theta_{v|u,C}(s_v, s_u, s_C) \tag{7.8}$$

And from here we can easily calculate $P(\mathcal{C} = s_C|M, \mathcal{V} = S)$ as

$$P(\mathcal{C} = s_C|M, \mathcal{V} = S) = \frac{P(\mathcal{V} = S, \mathcal{C} = s_C|\overline{E}^*, \Theta_{\overline{E}^*})}{\sum_{c\in C} P(\mathcal{V} = S, \mathcal{C} = c|\overline{E}^*, \Theta_{\overline{E}^*})} \tag{7.9}$$

The algorithmic description of the complete ELBMA TAN induction procedure shown in algorithm 8. The algorithm uses the maximum likelihood principle for determining the best structure and equation 3.9 to fix the parameters. We will name this learning algorithm STAN+BMA.

7.1.3 Computational complexity

The computational complexity of the `Construct-TAN` procedure in algorithm 1 is $\mathcal{O}((N + r^3) \cdot n^2)$ where $r = \max(\max_{i\in V} \#A_i, \#C)$. For the general `ELBMA` procedure appearing in algorithms 6 and 7 the costs are:

$$\begin{gathered} Cost(P(\mathcal{C} = s_C|M, \mathcal{V} = S)) = \mathcal{O}(n \cdot r) \\ Cost(\texttt{CalculateProb}) = \mathcal{O}(N \cdot n \cdot r) \\ Cost(\texttt{LBMA}) = \mathcal{O}(k \cdot N \cdot n \cdot r) \end{gathered} \tag{7.10}$$

where $k = \#\mathcal{M}'$, and n the number of attributes.
Knowing that:

$$\begin{gathered} Cost(Probability\ Approximation) = Cost(Counting) = \mathcal{O}((N + r^3) \cdot n^2) \\ Cost(Model\ Proposal) = \mathcal{O}((\log(\beta(n^2, n)) + k) \cdot n^2) \end{gathered} \tag{7.11}$$

we have that the total computational cost of the TAN induction algorithm that appears in algorithm 8 using BMA is

$$\begin{aligned} Cost(\texttt{LBMA-TAN}) &= Cost(Counting) + Cost(Model\ Proposal) + Cost(\texttt{LBMA}) \\ &= \mathcal{O}(N \cdot n(n + k \cdot r) + [r^3 + \log(\beta(n^2, n)) + k] \cdot n^2) \end{aligned} \tag{7.12}$$

which can be more easily understood as $\mathcal{O}(N \cdot n(n + k \cdot r) + (r^3 + k) \cdot n^2)$ for most practical purposes. This means that as long as we keep $\mathcal{M}'$ small, the

```
procedure LBMA-TAN (Dataset 𝒟)
    var
       ProbabilityDistribution P*_𝒟
       DirectedGraphSet 𝓜';
    begin
        Calculate P*_𝒟 by using equation 3.8;
        𝓜' = Construct-K-TAN(P*_𝒟,k);
        foreach M ∈ 𝓜'
            Set the weights of M according to equation 3.9;
        end
        return LBMA(𝒟,M');
    end

procedure Construct-K-TAN (ProbabilityDistribution P, Integer k)
    var
       WeightMatrix I_P;
       UndirectedGraph UG;
       UndirectedTreeSet UTS;
       DirectedTreeSet TS;
       DirectedGraphSet 𝓜';
    begin
        foreach A_i, A_j
            Compute I_P(A_i; A_j|C) as in Construct-TAN
        end
        G = ConstructUndirectedGraph(I_P);
        /* Returns the k maximum weighted spanning trees */
        UTS = K-MaximumWeightedSpanningTree(G,k);
        TS = MakeDirected(UTS);
        𝓜' = AddClass(TS);
        return 𝓜';
    end
```

Algorithm 8: ELBMA TAN learning procedure

computational overhead with respect to `Construct-TAN` will not be large. It grows linearly on the number of models and the number of learning instances.

After considering the learning time complexity, we should consider the classification time complexity. The cost of classifying a new instance with a single TAN model this is $\mathcal{O}(n \cdot r)$. With a multiple TAN model the cost is $O(k \cdot n \cdot r)$.

7.2 Experimental results

7.2.1 Adjusting the algorithm to run

To run the algorithm described in Section 7.1.2, we fixed k to 10. This value was chosen to show that you do not need to average over a large set of models in order to improve accuracy. We would like to point that k (and equivalently γ in the general version of LBMA) can act as an *effort knob*, in the sense of (Thearling, 1998), hence providing a useful feature for data mining users that allows them to decide how much computational power they want to spend in the task.

7.2.2 Experimental setting

We tested STAN and STAN+BMA over 17 datasets from the Irvine repository (Blake et al., 1998). The dataset characteristics are described in section 6.4.1. To discretize continuous attributes we used equal frequency discretization with 5 intervals. For each dataset and algorithm we tested both error rate and *LogScore* (as defined in section 6.4). For the evaluation of both error rate and *LogScore* we used 10 fold cross validation. We tested the algorithm with the 10%, 50% and 100% of the learning data for each fold, in order to get an idea of the influence of the amount of data in the behaviors of both error rate and *LogScore* for the algorithm.

The error rates appear in tables 7.1,7.3,7.5, with the best method for each dataset boldfaced. *LogScore*'s appear in tables 7.2,7.4,7.6. The columns of the tables are the induction methods and the rows are the datasets. The meaning of the column headers are:

- STAN is the softened TAN algorithm as described in (Friedman et al., 1997)
- STAN+BMA is the result of applying ELBMA directly to the STAN algorithm.

7.2.3 Interpretation of the results

Looking at the tables 7.1 - 7.6 it is easy to see that for most of the datasets, applying ELBMA improves both error rate and *LogScore*. Furthermore, after performing a 5% statistical significance t-test, we have that STAN+BMA error rate is significantly better than STAN 5, 5 and 6 times with 10%, 50% and 100% of the learning data respectively, whilst STAN error rate is never better than

Dataset	sTAN	sTAN+BMA
ADULT	**17.60 ± 0.82**	17.60 ± 0.80
AUSTRALIAN	25.39 ± 1.18	**24.96 ± 1.13**
BREAST	8.73 ± 0.87	**7.73 ± 0.93**
CAR	19.38 ± 0.95	**17.60 ± 0.77**
CHESS	**10.89 ± 0.56**	10.91 ± 0.53
CLEVE	32.37 ± 1.00	**31.89 ± 1.27**
CRX	25.14 ± 0.87	**24.18 ± 0.98**
FLARE	19.94 ± 0.85	**19.92 ± 0.88**
GLASS	59.19 ± 1.78	**58.54 ± 1.83**
GLASS2	37.75 ± 1.39	**36.63 ± 1.37**
IRIS	25.87 ± 3.07	**24.80 ± 2.96**
LETTER	36.11 ± 1.39	**34.68 ± 1.37**
LIVER	42.39 ± 0.94	**41.24 ± 1.37**
NURSERY	8.88 ± 1.12	**8.50 ± 1.12**
PRIMARY-TUMOR	**71.67 ± 1.54**	71.73 ± 1.44
SOYBEAN	**30.79 ± 1.28**	30.82 ± 1.33
VOTES	14.14 ± 0.93	**14.13 ± 0.71**

Table 7.1: Averages and standard deviations of error rate using 10% of the learning data

Dataset	sTAN	sTAN+BMA
ADULT	**567.09 ± 3.92**	567.64 ± 4.00
AUSTRALIAN	17.85 ± 0.64	**17.06 ± 0.60**
BREAST	8.12 ± 0.69	**7.56 ± 0.65**
CAR	38.55 ± 0.91	**36.52 ± 0.86**
CHESS	**35.39 ± 0.58**	35.40 ± 0.59
CLEVE	8.49 ± 0.74	**8.23 ± 0.76**
CRX	17.84 ± 1.05	**16.89 ± 1.00**
FLARE	24332.38 ± 56.59	**24332.03 ± 56.59**
GLASS	11713.24 ± 72.91	**11713.00 ± 72.91**
GLASS2	4.68 ± 0.57	**4.57 ± 0.54**
IRIS	4.04 ± 0.67	**3.96 ± 0.70**
LETTER	1385.73 ± 8.95	**1300.23 ± 8.38**
LIVER	12.62 ± 0.79	**11.71 ± 0.65**
NURSERY	3126.39 ± 77.45	**3123.62 ± 77.45**
PRIMARY-TUMOR	75927.03 ± 123.39	**75926.94 ± 123.39**
SOYBEAN	41125.59 ± 108.25	**41125.46 ± 108.25**
VOTES	6.09 ± 0.50	**6.03 ± 0.48**

Table 7.2: Averages and standard deviations of *LogScore* using 10% of the learning data

Dataset	sTAN	sTAN+BMA
ADULT	16.46 ± 0.78	**16.45 ± 0.83**
AUSTRALIAN	18.14 ± 0.91	**17.74 ± 0.80**
BREAST	5.26 ± 0.84	**4.75 ± 0.72**
CAR	8.68 ± 0.68	**8.09 ± 0.58**
CHESS	8.25 ± 0.49	**8.15 ± 0.49**
CLEVE	24.01 ± 1.31	**23.57 ± 1.28**
CRX	18.12 ± 0.92	**17.68 ± 0.85**
FLARE	18.55 ± 0.62	**18.54 ± 0.72**
GLASS	**33.79 ± 1.14**	33.86 ± 0.97
GLASS2	**22.38 ± 1.53**	23.40 ± 1.54
IRIS	8.40 ± 1.00	**8.27 ± 0.82**
LETTER	15.62 ± 0.91	**15.31 ± 0.83**
LIVER	36.73 ± 1.60	**35.17 ± 1.34**
NURSERY	7.09 ± 0.80	**6.03 ± 0.97**
PRIMARY-TUMOR	60.23 ± 1.17	**59.87 ± 1.33**
SOYBEAN	7.88 ± 0.71	**7.80 ± 0.82**
VOTES	**7.63 ± 0.93**	7.76 ± 0.93

Table 7.3: Averages and standard deviations of error rate using 50% of the learning data

Dataset	sTAN	sTAN+BMA
ADULT	520.03 ± 3.93	**518.82 ± 3.91**
AUSTRALIAN	14.79 ± 0.76	**14.41 ± 0.59**
BREAST	5.17 ± 0.64	**4.40 ± 0.62**
CAR	20.44 ± 0.51	**19.73 ± 0.48**
CHESS	27.32 ± 0.73	**27.12 ± 0.79**
CLEVE	7.38 ± 0.66	**7.15 ± 0.63**
CRX	15.62 ± 1.11	**15.21 ± 1.07**
FLARE	4233.42 ± 41.82	**4233.31 ± 41.82**
GLASS	309.52 ± 24.49	**309.25 ± 24.49**
GLASS2	3.86 ± 0.53	**3.68 ± 0.51**
IRIS	1.52 ± 0.37	**1.48 ± 0.34**
LETTER	574.47 ± 6.13	**559.56 ± 6.17**
LIVER	10.78 ± 0.74	**10.39 ± 0.71**
NURSERY	1596.96 ± 67.06	**1594.32 ± 67.06**
PRIMARY-TUMOR	12028.93 ± 51.79	**12028.74 ± 51.79**
SOYBEAN	907.34 ± 42.43	**907.27 ± 42.43**
VOTES	5.04 ± 0.80	**4.50 ± 0.69**

Table 7.4: Averages and standard deviations of *LogScore* using 50% of the learning data

Dataset	sTAN	sTAN+BMA
ADULT	16.46 ± 0.68	**16.42 ± 0.72**
AUSTRALIAN	16.49 ± 0.65	**16.43 ± 0.72**
BREAST	4.29 ± 0.66	**3.72 ± 0.45**
CAR	6.23 ± 0.55	**6.16 ± 0.53**
CHESS	7.89 ± 0.38	**7.68 ± 0.44**
CLEVE	19.99 ± 1.26	**19.73 ± 1.18**
CRX	**15.71 ± 0.66**	15.79 ± 0.74
FLARE	18.46 ± 0.30	**18.31 ± 0.24**
GLASS	26.58 ± 1.22	**25.99 ± 1.28**
GLASS2	19.61 ± 1.42	**18.06 ± 1.43**
IRIS	8.13 ± 1.44	**7.20 ± 1.43**
LETTER	12.69 ± 0.77	**12.48 ± 0.83**
LIVER	33.36 ± 0.98	**33.19 ± 1.10**
NURSERY	6.62 ± 0.75	**4.81 ± 0.76**
PRIMARY-TUMOR	56.74 ± 1.09	**56.32 ± 0.93**
SOYBEAN	5.97 ± 0.50	**5.94 ± 0.49**
VOTES	**6.26 ± 0.81**	6.34 ± 0.56

Table 7.5: Averages and standard deviations of error rate using 100% of the learning data

Dataset	sTAN	sTAN+BMA
ADULT	508.10 ± 3.07	**508.01 ± 3.07**
AUSTRALIAN	12.90 ± 0.65	**12.66 ± 0.61**
BREAST	4.85 ± 0.50	**4.28 ± 0.54**
CAR	**16.29 ± 0.39**	16.31 ± 0.41
CHESS	26.46 ± 0.46	**26.22 ± 0.36**
CLEVE	6.51 ± 0.44	**6.29 ± 0.51**
CRX	13.97 ± 0.68	**13.76 ± 0.58**
FLARE	1532.39 ± 0.62	**1532.22 ± 0.65**
GLASS	7.40 ± 0.59	**7.12 ± 0.52**
GLASS2	3.20 ± 0.39	**3.08 ± 0.37**
IRIS	1.18 ± 0.44	**1.16 ± 0.44**
LETTER	441.94 ± 5.61	**433.37 ± 5.84**
LIVER	**9.59 ± 0.44**	9.72 ± 0.60
NURSERY	91.52 ± 2.41	**89.41 ± 2.30**
PRIMARY-TUMOR	6327.87 ± 38.33	**6327.64 ± 38.33**
SOYBEAN	4.49 ± 0.51	**4.45 ± 0.48**
VOTES	3.96 ± 0.55	**3.76 ± 0.46**

Table 7.6: Averages and standard deviations of *LogScore* using 100% of the learning data

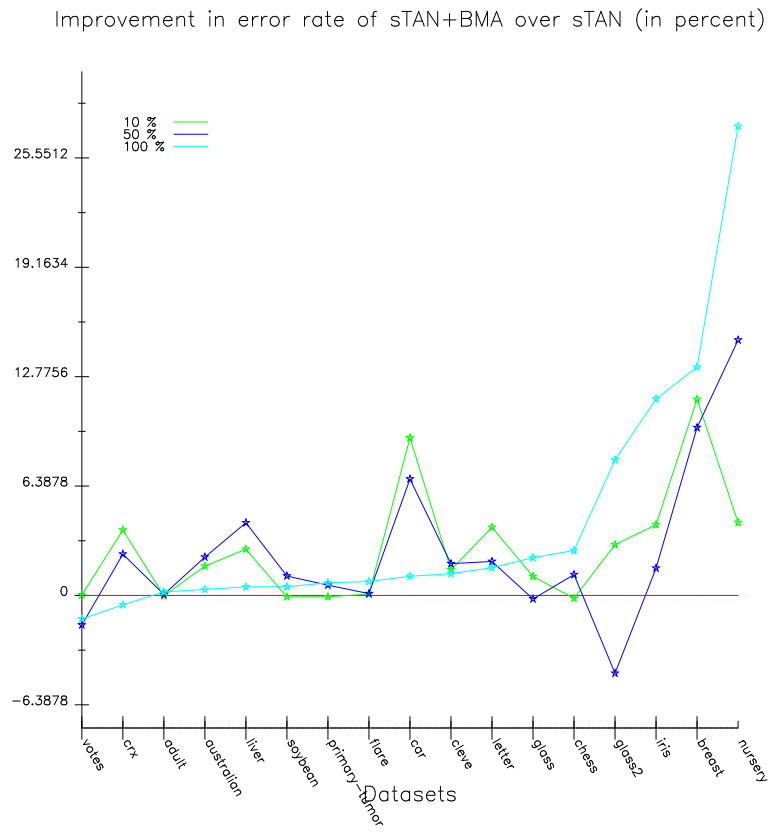


Figure 7.1: Comparison of STAN+BMA and STAN error rate

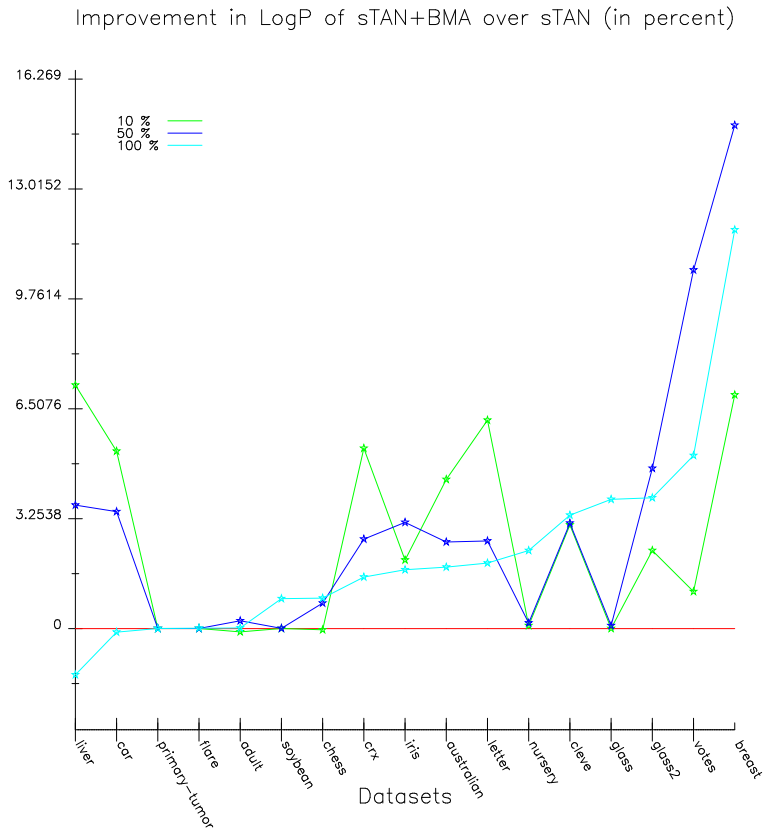


Figure 7.2: Comparison of sTAN+BMA and sTAN *LogScore*

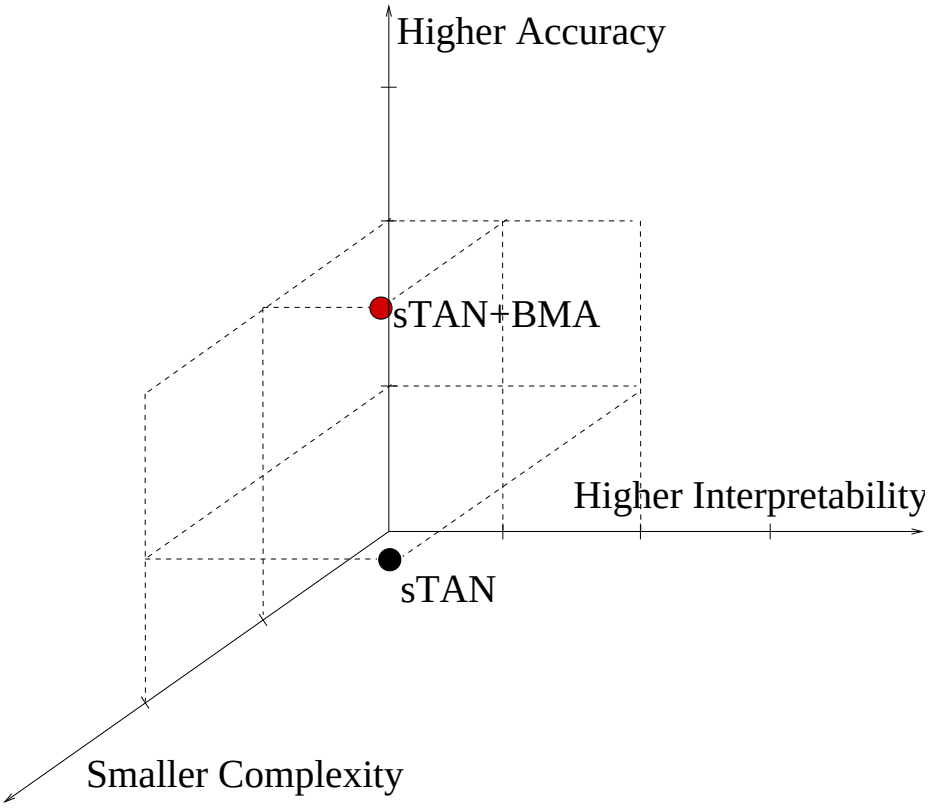


Figure 7.3: Relative positioning of the STAN+BMA and STAN

STAN+BMA in a statistically significant way. For *LogScore* results are even more conclusive. STAN+BMA *LogScore* is significantly better than STAN 12, 14 and 12 times with 10%, 50% and 100% of the learning data respectively, whilst STAN *LogScore* is better than STAN+BMA in a statistically significant way only for one dataset and only with 100% of the training data. The difference between both classifiers is plotted in figures 7.1 and 7.2.

7.3 Summary

In this chapter we have provided a solution to the fact that the TAN induction algorithm presented in (Friedman et al., 1997) does not account for uncertainty in model selection. We have introduced both *local Bayesian model averaging* and *empirical local Bayesian model averaging* as techniques to deal with uncertainty in model selection that can be applied without too many additional requirements to algorithms inducing probabilistic classifiers. We have seen how *empirical local*

Bayesian model averaging can be applied for TAN induction. We have provided empirical evidence that shows that in most of the cases the resulting new method provides more accurate predictions and probability estimates.

We have tried to summarize in Figure 7.3 the interpretation of the results in this chapter in terms of the three dimensional space for classifiers defined in the introduction. There we can see that STAN+BMA improves considerably STAN accuracy at the cost of increasing complexity and decreasing understandability.

7.3.1 Further research

An analysis of the final weights of the different models suggests that it makes sense to take an adaptive approach to ELBMA, starting with a large number of models and reducing it progressively as we accumulate evidence "against them" (some of the weights were of the order of 10^{-20} even for not very large datasets).

In the next two chapters we show that under suitable conditions, the Bayesian model averaging integral for TANs can be calculated in closed form in polynomial time (chapter 8) and that it is possible to efficiently find the k structures with maximum a posteriori probability (chapter 9). This last result will provide a classifier very similar in spirit to STAN+BMA but with a significantly reduced learning time complexity and an improved theoretical basis.

Chapter 8

Tractable Bayesian Model Averaging of Tree Augmented Naive Bayes Classifiers

Give no decision till both sides thou'st heard.

Phocylides

In the previous chapter we have seen that the TAN induction algorithm presented in (Friedman et al., 1997) can be improved by taking into account model uncertainty. We used empirical local Bayesian model averaging to approximate Bayesian model averaging because of the computational cost of numerically calculating the exact result.

In this chapter we show that, under suitable assumptions, the Bayesian model averaging of TAN can be integrated in closed form and that it leads to improved classification performance. The chapter is organized as follows. In section 8.1 we review and correct the results in (Meila and Jaakkola, 2000b). The proofs for the results in this and the next section are provided in appendix B. In section 8.2 we develop the closed expression for the Bayesian model averaging of TAN and we construct a classifier based on this result which we will name TBMATAN (from Tractable Bayesian Model Averaging of Tree Augmented Naive-Bayes). In section 8.3 we notice that TBMATAN has a major drawback that makes difficult its usage for large datasets. The drawback comes from the fact that the classifier depends on the calculation of an ill-conditioned determinant. The computation of this determinant requires the floating point precision to increase with the dataset size and hence increases the computing time. To solve this

drawback we introduce SSTBMATAN, an approximation of TBMATAN. In section 8.4 we study the empirical characteristics of TBMATAN and show that it leads to a reduced error rate and to a better approximation of the class probabilities with respect to STAN. We also show that the empirical results for SSTBMATAN significantly improve the ones obtained by TBMATAN. This, together with the fact that SSTBMATAN can deal with large datasets position SSTBMATAN as an interesting classifier. As usual, we end up with some conclusions and future work in section 8.5.

8.1 Decomposable distributions over tree belief networks

8.1.1 Definition

In (Meila and Jaakkola, 2000b), Meila and Jaakkola introduced *decomposable priors*: a family of priors over structure and parameters of tree belief networks, that is, a family of probability distributions over the space of tree belief network models. In the following we will use the term decomposable distribution over trees instead of decomposable priors, and we will use prior and posterior as they are commonly used in Bayesian analysis.

Decomposable distributions over trees are the product of a distribution over tree structures and a distribution over tree parameters, that is, assuming that given ξ the probability distribution over the set of tree belief network models follows a decomposable distribution we have that

$$P(M|\xi) = P(\overline{E}, \Theta_{\overline{E}}|\xi) = P(\overline{E}|\xi)P(\Theta_{\overline{E}}|\overline{E}, \xi) \tag{8.1}$$

where we recall that $\overline{E}$ is the directed tree structure and $\Theta_{\overline{E}}$ its parameters. The definition of decomposable distribution will be done by specifying its two components, $P(\overline{E}|\xi)$, the decomposable distribution over tree structures and $P(\Theta_{\overline{E}}|\overline{E}, \xi)$, the decomposable distribution over tree parameters.

Decomposable distributions over tree structures

Recalling that m is the number of variables when talking about an unclassified discrete domain, a decomposable distribution over tree structures is determined by a $m \times m$ hyperparameter matrix $\boldsymbol{\beta} = (\beta_{u,v})$ such that $\forall u, v : 1 \leq u, v \leq m$: $\beta_{u,v} = \beta_{v,u} \geq 0$; $\beta_{v,v} = 0$. We can interpret $\beta_{u,v}$ as a measure of how possible is under ξ that the edge $(\mathcal{X}_u, \mathcal{X}_v)$ is contained in the tree model underlying the data.

We say that $P(\overline{E}|\xi)$ follows a decomposable distribution over tree structures with hyperparameter set $\boldsymbol{\beta}$ iff:

$$P(\overline{E}|\xi) = \frac{P(E|\xi)}{m} \tag{8.2}$$

$$P(E|\xi) = \frac{1}{Z_\beta} \prod_{u,v \in E} \beta_{u,v} \tag{8.3}$$

where Z_β is a normalization constant with value:

$$Z_\beta = \sum_{E \in \mathcal{E}} \prod_{u,v \in E} \beta_{u,v} \tag{8.4}$$

It is worth noting that $P(\overline{E}|\xi)$ depends only on the underlying undirected tree structure E.

Decomposable distributions over tree parameters

A decomposable distribution over tree parameters follows the equation

$$P(\Theta_{\overline{E}}|\overline{E},\xi) = P(\boldsymbol{\theta}_{\rho_{\overline{E}}}|\overline{E},\xi) \prod_{u,v \in \overline{E}} P(\boldsymbol{\theta}_{v|u}|\overline{E},\xi) \tag{8.5}$$

where we remember that $\rho_{\overline{E}}$ is the root of $\overline{E}$. Furthermore, a decomposable distribution over tree parameters has a hyperparameter set $\mathbf{N}' = \{N'_{v,u}(j,i)|1 \leq u \neq v \leq m \ ; \ j \in X_v \ ; \ i \in X_u\}$ with the constraint that exist $N'_u(i)$, N' such that for every u,v:

$$N'_u(i) = \sum_{j \in X_v} N'_{v,u}(j,i) \tag{8.6}$$

$$N' = \sum_{i \in X_u} N'_u(i) \tag{8.7}$$

We say that $P(\Theta_{\overline{E}}|\overline{E},\xi)$ follows a decomposable distribution over tree parameters with hyperparameter set $\mathbf{N}'$ iff

- $P(\Theta_{\overline{E}}|\overline{E},\xi)$ fulfills equation 8.5
- $\mathbf{N}'$ fulfills the conditions appearing in equations 8.6, 8.7.
- the following two equations are also satisfied

$$P(\boldsymbol{\theta}_{\rho_{\overline{E}}}|\overline{E},\xi) = D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.)) \tag{8.8}$$

$$P(\boldsymbol{\theta}_{v|u}|\overline{E},\xi) = \prod_{i \in X_u} D(\theta_{v|u}(.,i); N'_{v,u}(.,i)) \tag{8.9}$$

  where D stands for the Dirichlet distribution introduced in section 3.2.1.

Decomposable distributions over tree structures and parameters

We say that $P(M|\xi)$ follows a decomposable distribution over trees with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$ iff

- $P(M|\xi)$ fulfills equation 8.1

- $P(\overline{E}|\xi)$ follows a decomposable distribution over tree structures with hyperparameter set $\boldsymbol{\beta}$
- $P(\Theta_{\overline{E}}|\overline{E},\xi)$ follows a decomposable distribution over tree parameters with hyperparameter set $\mathbf{N}'$

that is if the conditions in equations 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8 and 8.9 hold.

8.1.2 Meila and Jaakkola results and corrections to their results

We have reviewed decomposable distributions over trees. In (Meila and Jaakkola, 2000b), some important results about such distributions are stated. Unfortunately, some of these results were not stated correctly. We review those results and provide corrected versions when they are needed.

Assumptions needed for decomposable distributions over tree parameters

Meila and Jaakkola demonstrated that if we have a distribution over tree parameters satisfying equation 8.1, for which the support graph is connected and its parameter set is strictly positive, then under the assumptions of likelihood equivalence, parameter independence, parameter modularity and connectivity, for any tree in any directed representation, the parameters are distributed following a set of Dirichlets as imposed by equations 8.5,8.8,8.9,8.6 and 8.7. This means that decomposable distributions over tree parameters are the result of a fairly reasonable set of assumptions widely used for learning Bayesian networks as can be seen in (Heckerman et al., 1995).

Bayesian learning with decomposable distributions over trees

Meila and Jaakkola claimed that if we assume a decomposable distribution over trees with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$ the posterior distribution after a dataset $\mathcal{D}$ follows a decomposable distribution over trees with hyperparameters given by

$$\beta^*_{u,v} = \beta_{u,v} W_{u,v} \tag{8.10}$$

$$N'^*_{u,v}(j,i) = N'_{u,v}(j,i) + N_{u,v}(j,i) \tag{8.11}$$

where

$$W_{u,v} = \prod_{i \in X_u} \prod_{j \in X_v} \frac{\Gamma(N'_{v,u}(j,i) + N_{v,u}(j,i))}{\Gamma(N'_{v,u}(j,i))} \tag{8.12}$$

Corrected Bayesian learning with decomposable distributions over trees

Unfortunately, the last result is mistaken. After careful derivation, it can be proven that if we assume a decomposable prior distribution over trees with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$ the posterior distribution after a dataset $\mathcal{D}$ follows a decomposable distribution over trees with hyperparameters given by equations 8.10 and 8.11 but $W_{u,v}$ are given by:

$$\begin{aligned} W_{u,v} = & \prod_{i \in X_u} \frac{\Gamma(N'_u(i))}{\Gamma(N'_u(i) + N_u(i))} \\ & \prod_{j \in X_v} \frac{\Gamma(N'_v(j))}{\Gamma(N'_v(j) + N_v(j))} \\ & \prod_{i \in X_u} \prod_{j \in X_v} \frac{\Gamma(N'_{v,u}(j,i) + N_{v,u}(j,i))}{\Gamma(N'_{v,u}(j,i))} \end{aligned} \tag{8.13}$$

The demonstration can be seen in appendix B.2.2.

Computation of probabilities from the posterior

Meila and Jaakkola claimed that if we assume a decomposable prior distribution over trees with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$ the posterior probability of a new data point conditioned to the observation of a dataset $\mathcal{D}$ comes given by

$$P(\mathcal{X} = x | \mathcal{D}, \xi) = \frac{w_0(x) |Q(\boldsymbol{\beta}\ \mathbf{w}(x))|}{|Q(\boldsymbol{\beta}\mathbf{W})|} \tag{8.14}$$

where

$$w_0(x) = \frac{1}{N' + N} \prod_{X_u \in V} [N'_{pa(u)}(x_{pa(u)}) + N_{pa(u)}(x_{pa(u)})] \tag{8.15}$$

$$\mathbf{w}(x) = (w_{u,v}(x)) \text{ where } w_{u,v}(x) = \frac{N'_{v,u}(s_v, s_u) + N_{v,u}(s_v, s_u)}{(N'_u(s_u) + N_u(s_u))(N'_v(s_v) + N_v(s_v))} \tag{8.16}$$

and for any real $m \times m$ matrix $\boldsymbol{\tau}$ we define $\underline{Q}(\boldsymbol{\tau}) : \mathbb{R}^{m \times m} \rightarrow \mathbb{R}^{m-1 \times m-1}$ as the first $m-1$ lines and columns of the matrix $\overline{Q}(\boldsymbol{\tau})$ where

$$\overline{Q}_{u,v}(\boldsymbol{\tau}) = \overline{Q}_{v,u}(\boldsymbol{\tau}) = \begin{cases} -\tau_{u,v} & 1 \leq u < v \leq m \\ \sum_{v'=1}^{m} \tau_{v',v} & 1 \leq u = v \leq m \end{cases} \tag{8.17}$$

Corrected computation of probabilities from a decomposable distribution over trees

The previous result is also incorrect since they assume $\mathbf{W}$ defined as in equation 8.12. Furthermore, they also claim that $w_0(x)$ is a structure independent factor.

It is easy to see that this is not the case. If our domains contains two attributes, namely X_1 and X_2, for the tree $X_1 \rightarrow X_2$ we have that $w(x) = N'_1(x_1) + N_1(x_1)$ while for the tree $X_2 \rightarrow X_1$ we have that $w(x) = N'_2(x_2) + N_2(x_2)$. In fact, since we have seen that the posterior is also a decomposable distribution over trees, we think it is simpler to have a result regarding the probability of a new data point given a decomposable distribution over trees, since such a result can be applied to any decomposable distribution over trees, including the posterior. Hence, we state the corrected result as follows. Assume that $P(M|\xi)$ follows a decomposable distribution over trees with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$. The probability of an observation x given ξ comes given by

$$P(\mathcal{X} = x|\xi) = h_0^x |Q(\boldsymbol{\beta}\ \mathbf{h}^x)| \tag{8.18}$$

where

$$h_0^x = \frac{1}{Z_\beta}\frac{1}{N'}\prod_{X_u \in V} N'_u(x_u) \tag{8.19}$$

$$\mathbf{h}^x = \left(h^x_{u,v}\right) \text{ where } h^x_{u,v} = \frac{N'_{v,u}(x_v, x_u)}{N'_u(x_u)N'_v(x_v)} \tag{8.20}$$

The proof for this result appears in appendix B.2.1.

It is easy to see that it can be particularized for the posterior by using the corrected result for Bayesian learning with decomposable distributions over trees.

8.2 Development of the Averaged Tree Augmented Naive Bayes

In the previous section we have reviewed and corrected the results in (Meila and Jaakkola, 2000b) for trees. In this section we will extend them to TAN models. We will develop a classifier based on the TAN model that does also take into account the uncertainty in model selection by means of decomposable distributions over TANs. We start by introducing decomposable distributions over TAN structures and parameters, built upon the already presented idea of decomposable priors over trees. After that we demonstrate that given a decomposable distribution over TANs it is possible to calculate the probability of an unseen observation and that given a prior decomposable distribution over TANs, the posterior distribution after observing a set of data is also a decomposable distribution over TANs. We conclude the section by putting together these results to create TBMATAN.

8.2.1 Decomposable distributions over TANs

In this section we introduce decomposable distributions over TANs, which are a family of probability distributions in the space $\mathcal{M}$ of TAN models with a development parallel to the one for decomposable distributions over trees. Decomposable distributions over trees are based on four assumptions: likelihood

equivalence, parameter independence, parameter modularity and connectivity. These four assumptions are also the basis of the development of decomposable distributions over TANs. Specially significant, in order to understand the developments is likelihood equivalence. This assumption states that in all possible parameterizations consistent with a given undirected tree E the distribution will assign the same probability mass to any measurable subset in parameter space. This provides us with a very valuable tool when integrating over parameters, because it allows us to integrate over the parameters of any directed tree $\overline{E}$ obtained from E because for all of them the result of the integration should be the same.

Decomposable distributions over TANs are constructed in two steps. In the first step, a distribution over the set of different undirected tree structures is defined. Every directed tree structure is defined to have the same probability as its undirected equivalent. In the second step, a distribution over the set of parameters is defined so that is also independent on the structure. If $P(M|\xi)$ follows a decomposable distribution over TANs then the probability for a model $M = \langle \overline{E}^*, \Theta_{\overline{E}^*} \rangle$ (a TAN with fixed tree structure $\overline{E}^*$ and fixed parameters $\Theta_{\overline{E}^*}$) is determined by:

$$P(M|\xi) = P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi) = P(\overline{E}^*|\xi)P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi) \tag{8.21}$$

In the following sections we specify the value of a decomposable distribution over the two components of a TAN model, namely its structure and its parameters. That is, $P(\overline{E}^*|\xi)$ (decomposable distribution over TAN structures) and $P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi)$ (decomposable distribution over TAN parameters).

Decomposable distributions over TAN structures

Recalling that n is the number of attributes when talking about a classified discrete domain, a decomposable distribution over TAN structures is determined by an $n \times n$ hyperparameter matrix $\boldsymbol{\beta} = (\beta_{u,v})$ such that $\forall u, v : 1 \leq u, v \leq n \ : \beta_{u,v} = \beta_{v,u} \geq 0 \ ; \ \beta_{v,v} = 0$. We can interpret $\beta_{u,v}$ as a measure of how possible is under ξ that the edge $(\mathcal{A}_u, \mathcal{A}_v)$ is contained in the TAN model underlying the data.

We say that $P(\overline{E}^*|\xi)$ follows a decomposable distribution over TAN structures with hyperparameter set $\boldsymbol{\beta}$ iff:

$$P(\overline{E}^*|\xi) = \frac{P(E|\xi)}{n} \tag{8.22}$$

$$P(E|\xi) = \frac{1}{Z_\beta} \prod_{u,v \in E} \beta_{u,v} \tag{8.23}$$

where Z_β is a normalization constant with value:

$$Z_\beta = \sum_{E \in \mathcal{E}} \prod_{u,v \in E} \beta_{u,v} \tag{8.24}$$

It is worth noting that $P(\overline{E}^*|\xi)$ depends only on the underlying undirected tree structure E.

Decomposable distributions over TAN parameters

A decomposable distribution over TAN parameters follows the equation

$$P(\Theta_{\overline{E}^*}|\overline{E}^*,\xi) = P(\boldsymbol{\theta}_C|\overline{E}^*,\xi)\ P(\boldsymbol{\theta}_{\rho_{\overline{E}}|C}|\overline{E}^*,\xi)\ \prod_{u,v\in\overline{E}} P(\boldsymbol{\theta}_{v|u,C}|\overline{E}^*,\xi) \tag{8.25}$$

This can be interpreted as an application of equation 3.1 for the case of TAN. Furthermore, a decomposable distribution over TAN parameters has a hyperparameter set $\mathbf{N}' = \{N'_{v,u,C}(j,i,c)|1 \leq u \neq v \leq n\ ;\ j \in A_v\ ;\ i \in A_u\ ;\ c \in C\}$ with the constraint that exist $N'_{u,C}(i,c)$, $N'_C(c)$, N' such that for every u,v:

$$N'_{u,C}(i,c) = \sum_{j\in A_v} N'_{v,u,C}(j,i,c) \tag{8.26}$$

$$N'_C(c) = \sum_{i\in A_u} N'_{u,C}(i,c) \tag{8.27}$$

$$N' = \sum_{c\in C} N'_C(c) \tag{8.28}$$

We say that $P(\Theta_{\overline{E}^*}|\overline{E}^*,\xi)$ follows a decomposable distribution over TAN parameters with hyperparameter set $\mathbf{N}'$ iff

- $P(\Theta_{\overline{E}^*}|\overline{E}^*,\xi)$ fulfills equation 8.25
- $\mathbf{N}'$ fulfills the conditions appearing in equations 8.26, 8.27 and 8.28
- the following equations are also satisfied:

$$P(\boldsymbol{\theta}_C|\overline{E}^*,\xi) = D(\theta_C(.); N'_C(.)) \tag{8.29}$$

$$P(\boldsymbol{\theta}_{\rho_{\overline{E}}|C}|\overline{E}^*,\xi) = \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)) \tag{8.30}$$

$$P(\boldsymbol{\theta}_{v|u,C}|\overline{E}^*,\xi) = \prod_{c\in C}\prod_{i\in A_u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)) \tag{8.31}$$

Decomposable distributions over TAN structures and parameters

We say that $P(M|\xi)$ follows a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$ iff

- $P(M|\xi)$ fulfills equation 8.21
- $P(\overline{E}^*|\xi)$ follows a decomposable distribution over TAN structures with hyperparameter set $\boldsymbol{\beta}$
- $P(\Theta_{\overline{E}^*}|\overline{E}^*,\xi)$ follows a decomposable distribution over TAN parameters with hyperparameter set $\mathbf{N}'$

that is if the conditions in equations 8.21, 8.22, 8.23, 8.24, 8.25, 8.26, 8.27, 8.28, 8.29, 8.30 and 8.31 hold.

8.2.2 Calculating probabilities under decomposable distributions over TANs

Assume that the data is generated by a TAN model and that $P(M|\xi)$ follows a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$. We can calculate the probability of an observation S, s_C given ξ by averaging over the set of TAN models

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\xi) = \int_{M \in \mathcal{M}} P(\mathcal{V} = S, \mathcal{C} = s_C|M) P(M|\xi) \tag{8.32}$$

The integral for $P(\mathcal{V} = S, \mathcal{C} = s_C|\xi)$ can be calculated in closed form by applying the matrix tree theorem (see Appendix B.1.1 and (Meila and Jaakkola, 2000b)) and expressed in terms of the previously introduced Q as:

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\xi) = h_0^{S,s_C} |Q(\boldsymbol{\beta}\ \mathbf{h}^{S,s_C})| \tag{8.33}$$

where

$$h_0^{S,s_C} = \frac{1}{Z_\beta} \frac{1}{N'} \prod_{A_u \in V} N'_{u,C}(s_u, s_C) \tag{8.34}$$

$$\mathbf{h}^{S,s_C} = \left(h_{u,v}^{S,s_C}\right) \text{ where } h_{u,v}^{S,s_C} = \frac{N'_{v,u,C}(s_v, s_u, s_C)}{N'_{u,C}(s_u, s_C) N'_{v,C}(s_v, s_C)} \tag{8.35}$$

The proof for this result appears in appendix B.3.1.

8.2.3 Learning under decomposable distributions over TANs

Assume that the data is generated by a TAN model and that $P(M|\xi)$ follows a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}$, $\mathbf{N}'$. Then, $P(M|\mathcal{D}, \xi)$, the posterior probability distribution after observing a dataset $\mathcal{D}$, containing independent identically distributed observations, is a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}^*$, $\mathbf{N}'^*$ given by:

$$\beta^*_{u,v} = \beta_{u,v} W_{u,v} \tag{8.36}$$

$$N'^*_{u,v,C}(j, i, c) = N'_{u,v,C}(j, i, c) + N_{u,v,C}(j, i, c) \tag{8.37}$$

where

$$\begin{aligned} W_{u,v} = & \prod_{c \in C} \prod_{i \in A_u} \frac{\Gamma(N'_{u,C}(i, c))}{\Gamma(N'_{u,C}(i, c) + N_{u,C}(i, c))} \\ & \prod_{c \in C} \prod_{j \in A_v} \frac{\Gamma(N'_{v,C}(j, c))}{\Gamma(N'_{v,C}(j, c) + N_{v,C}(j, c))} \\ & \prod_{c \in C} \prod_{i \in A_u} \prod_{j \in A_v} \frac{\Gamma(N'_{v,u,C}(j, i, c) + N_{v,u,C}(j, i, c))}{\Gamma(N'_{v,u,C}(j, i, c))} \end{aligned} \tag{8.38}$$

The proof appears in appendix B.3.2.

8.2.4 Putting it all Together

Putting together the results from sections 8.2.2 and 8.2.3 we can easily design a classifier based on decomposable distributions over TANs. The classifier works as follows: when given a dataset $\mathcal{D}$, it assumes that the data is generated from a TAN model and assumes a decomposable distribution over TANs as prior over the set of models. Applying the result from section 8.2.3, the posterior distribution over the set of models is also a decomposable distribution over TANs and applying the result of section 8.2.2 this decomposable posterior distribution over TANs can be used to calculate the probability of any observation S, s_C. When given an unclassified observation S, it can just calculate the probability $P(\mathcal{V} = S, \mathcal{C} = s_C|\mathcal{D}, \xi)$ for each possible class $s_C \in C$ and classify S in the class with highest probability.

We have mentioned that the classifier assumes a decomposable distribution over TANs as prior. Ideally, this prior will be fixed by an expert that knows the classification domain. Otherwise, we have to provide the classifier with a way for fixing the prior distribution hyperparameters without knowledge about the domain. In this case the prior should be as "non-informative" as possible in order for the information coming from $\mathcal{D}$ to dominate the posterior by the effects of equations 8.36 and 8.37. We have translated this requisite into equations 8.39 and 8.40:

$$\forall u, v \ ; \ 1 \leq u \neq v \ \leq n \ ; \ \beta_{u,v} = 1 \tag{8.39}$$

$$\forall u, v; 1 \leq u \neq v \leq n; \forall j \in A_v; \forall i \in A_u; \forall c \in C; N'_{v,u,C}(j, i, c) = \frac{\lambda}{\#C\#A_u\#A_v} \tag{8.40}$$

Defining $\boldsymbol{\beta}$ as in equation 8.39 means that we have the same amount of belief for any edge being in the TAN structure underlying the data. Fixed u, v, equation 8.40 assigns the same probability to any (j, i, c) such that $j \in A_v$, $i \in A_u$ and $c \in C$ λ is an "equivalent sample size" for the prior in the sense of Heckerman et al. (Heckerman et al., 1995). In our experiments we have fixed $\lambda = 10$. In the following TBMATAN will refer to the classifier described in this section.

8.3 Approximating TBMATAN

8.3.1 TBMATAN computational complexity

The learning time complexity for TBMATAN is bounded by the counting step, that is, $\mathcal{O}((N + r^3) \cdot n^2)$ where $r = \max(\max_{i \in V} \#A_i, \#C)$. Once the learning step is over, the classification time complexity requires the computation of $\#C < r$ determinants. Since computing the determinant of a $n \times n$ matrix is $\mathcal{O}(n^3)$, the classification time complexity for TBMATAN is bounded by $\mathcal{O}(n^3 \cdot r)$.

8.3.2 Computational problems

A straightforward implementation of TBMATAN, even when accomplishing the beforementioned complexity bounds, will not yield accurate results, specially for large datasets. This is due to the fact that the calculations that need to be done in order to classify a new observation include the computation of a determinant (in equation 8.33) that happens to be ill-conditioned. Even worse, the determinant gets more and more ill-conditioned as the number of observations in the dataset increases. This forces the floating point accuracy that we have to use to calculate these determinants to depend on the dataset size. We would like to note that this problem is due to the straightforward implementation of the formulas. If it were possible to compute quotients of determinants of similar matrixes accurately, the problem would be solved. To the best of our knowledge, such accurate computation does not exist. Therefore, we have used a brute force solution to accurately implement TBMATAN. More concretely, we have calculated the determinants by means of NTL (Shoup, 2003), a library that allows us to calculate determinants with the desired precision arithmetic. This solution makes the time for classifying a new observation grow faster than $\mathcal{O}(n^3 \cdot r)$, and hence makes the practical application of the algorithms difficult in situations where it is required to classify a large set of unclassified data.

8.3.3 A solution to TBMATAN computational problems

We analyzed what makes the determinant in TBMATAN being ill-conditioned and concluded that it is due to the $W_{u,v}$ factors given by equation 8.38. The factor $W_{u,v}$ could be interpreted as "how much the dataset $\mathcal{D}$ has changed the belief in that there is a link between u and v in the TAN model generating the data". The problems relies in the fact that $W_{u,v}$ are easily in the order of 10^{-200} for a dataset with 1500 observations. Furthermore, the factors $\frac{W_{u,v}}{W_{u',v'}}$ for such a dataset can be around 10^{-20}, providing the ill-condition of the determinant. In order to overcome this problem, we propose to postprocess the factors $W_{u,v}$ computed by equation 8.38 by means of a transformation that limits them to lie in the interval $[10^{-K}, 1]$ where K is a constant that has to be fixed depending on the floating point accuracy of the machine. In our implementation we have used $K = 5$.

The transformation works as depicted in Figure 8.1 and is described in detail by the following equations:

$$lmax = \log_{10} \max_{\substack{u \in V \\ v \in V \\ u \neq v}} W_{u,v} \tag{8.41}$$

$$lmin = \log_{10} \min_{\substack{u \in V \\ v \in V \\ u \neq v}} W_{u,v} \tag{8.42}$$

$$a = \begin{cases} \frac{K}{lmax - lmin} & lmax - lmin > K \\ 1 & otherwise \end{cases} \tag{8.43}$$

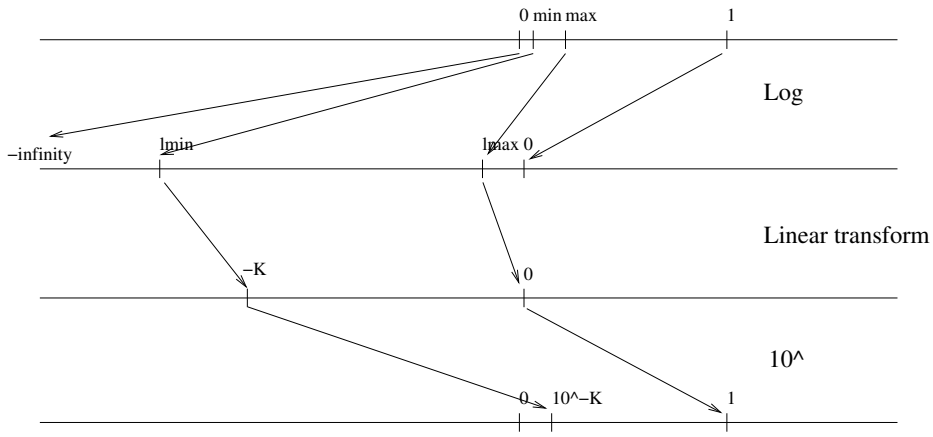


Figure 8.1: Transformation of weights for SSTBMATAN

$$b = -K - a * lmin \tag{8.44}$$

$$\widetilde{W}_{u,v} = 10^{a \log_{10}(W_{u,v})+b} \tag{8.45}$$

Using $\widetilde{W}_{u,v}$ instead of $W_{u,v}$ to calculate the posterior hyperparameters $\beta^*_{u,v}$ has the following properties:

1. It is harder to get get ill-conditioned determinants, because for all u, v $\widetilde{W}_{u,v}$ is bound to the interval $[10^{-K}, 1]$.

2. It preserves the relative ordering of the $W_{u,v}$. That is, if $W_{u,v} > W_{u',v'}$ then $\widetilde{W}_{u,v} > \widetilde{W}_{u',v'}$.

3. It does not exaggerate relative differences in belief. That is, for all u, v, u', v' we have that

   - If $\frac{W_{u,v}}{W_{u',v'}} \geq 1$ then $\frac{W_{u,v}}{W_{u',v'}} \geq \frac{\widetilde{W}_{u,v}}{\widetilde{W}_{u',v'}}$.
   - If $\frac{W_{u,v}}{W_{u',v'}} \leq 1$ then $\frac{W_{u,v}}{W_{u',v'}} \leq \frac{\widetilde{W}_{u,v}}{\widetilde{W}_{u',v'}}$.

The posterior hyperparameters $\beta^*_{u,v}$ can be interpreted as a representation of the a posteriori belief in the existence of an edge (u, v) in the TAN structure. Using $\widetilde{W}_{u,v}$, given the properties stated, means being more conservative in the structure learning process, because the beliefs will be confined to the interval $[10^{-K}, 1]$ which impedes the representation of extreme probability differences between edges. We can interpret the transformation as applying some stubbornness to the structure learning process. Applying this transformation allow us to implement an approximation of TBMATAN that does not require the use of special floating point accuracy computations. We will refer to this approximation of TBMATAN as SSTBMATAN (from Structure Stubborn TBMATAN).

It is worth noting that the problem described in this section does only affect the classification time complexity. The learning process for TBMATAN does not need high precision arithmetics. The learning time complexity for TBMATAN, $\mathcal{O}((N + r^3) \cdot n^2)$, is the same as the one for STAN and SSTBMATAN.

8.4 Empirical Results

We tested four algorithms over 17 datasets from the Irvine repository (Blake et al., 1998). The dataset characteristics are described in section 6.4.1. To discretize continuous attributes we used equal frequency discretization with 5 intervals. For each dataset and algorithm we tested both error rate and *LogScore* (see section 6.4). For the evaluation of both error rate and *LogScore* we used 10 fold cross validation. We tested the algorithm with the 10%, 50% and 100% of the learning data for each fold, in order to get an idea of the influence of the amount of data in the behaviors of both error rate and *LogScore* for the algorithm.

The error rates appear in Tables 8.1,8.3,8.5 with the best method for each dataset boldfaced. *LogScore*'s appear in Tables 8.2,8.4,8.6. The columns of the tables are the induction methods and the rows are the datasets. The meaning of the column headers are:

- SSTBMATAN is the method described in section 8.3.
- STAN is the softened TAN induction algorithm as presented in (Friedman et al., 1997).
- STAN+BMA is the TAN induction algorithm using ELBMA to deal with model uncertainty as presented in section 7.1.2.
- TBMATAN, is the method described in section 8.2.4.

TBMATAN classification times are very large for datasets with a large number of instances. For datasets over 5000 instances we have skipped the execution of TBMATAN. This is represented as a - sign in the corresponding table entries. We have skipped those datasets also when drawing comparison graphs for TBMATAN.

8.4.1 Interpretation of the Results

Summarizing the empirical results in the tables, we can conclude that:

- SSTBMATAN is a good approximation to TBMATAN even improving its results in some cases.
- SSTBMATAN improves STAN error rate and *LogScore*.
- SSTBMATAN improves STAN+BMA error rate and *LogScore*.

Dataset	SSTBMATAN	sTAN	sTAN+BMA	TBMATAN
ADULT	**16.73 ± 0.74**	17.60 ± 0.82	17.60 ± 0.80	-
AUSTRALIAN	**18.20 ± 1.11**	25.39 ± 1.18	24.96 ± 1.13	19.01 ± 1.03
BREAST	11.45 ± 1.22	8.73 ± 0.87	**7.73 ± 0.93**	14.97 ± 1.15
CAR	**16.08 ± 0.85**	19.38 ± 0.95	17.60 ± 0.77	16.51 ± 0.96
CHESS	11.74 ± 0.89	10.89 ± 0.56	10.91 ± 0.53	**9.27 ± 0.81**
CLEVE	**24.49 ± 1.12**	32.37 ± 1.00	31.89 ± 1.27	24.75 ± 1.07
CRX	**17.39 ± 0.98**	25.14 ± 0.87	24.18 ± 0.98	18.46 ± 0.96
FLARE	22.46 ± 0.94	19.94 ± 0.85	**19.92 ± 0.88**	22.76 ± 1.10
GLASS	**37.99 ± 1.52**	59.19 ± 1.78	58.54 ± 1.83	40.97 ± 1.20
GLASS2	**32.18 ± 2.18**	37.75 ± 1.39	36.63 ± 1.37	32.30 ± 2.15
IRIS	24.80 ± 2.13	25.87 ± 3.07	24.80 ± 2.96	**24.13 ± 2.13**
LETTER	**24.02 ± 0.93**	36.11 ± 1.39	34.68 ± 1.37	-
LIVER	43.51 ± 1.14	42.39 ± 0.94	**41.24 ± 1.37**	43.74 ± 1.17
NURSERY	**7.43 ± 0.98**	8.88 ± 1.12	8.50 ± 1.12	-
PRIMARY-TUMOR	**70.98 ± 1.77**	71.67 ± 1.54	71.73 ± 1.44	71.28 ± 1.73
SOYBEAN	**24.83 ± 1.14**	30.79 ± 1.28	30.82 ± 1.33	-
VOTES	8.13 ± 0.59	14.14 ± 0.93	14.13 ± 0.71	**8.09 ± 0.77**

Table 8.1: Averages and standard deviations of error rate using 10% of the learning data

Dataset	SSTBMATAN	sTAN	sTAN+BMA	TBMATAN
ADULT	**520.49 ± 3.40**	567.09 ± 3.92	567.64 ± 4.00	-
AUSTRALIAN	**14.28 ± 0.76**	17.85 ± 0.64	17.06 ± 0.60	16.57 ± 0.88
BREAST	11.09 ± 1.36	8.12 ± 0.69	**7.56 ± 0.65**	18.03 ± 1.54
CAR	**31.02 ± 0.93**	38.55 ± 0.91	36.52 ± 0.86	32.53 ± 0.97
CHESS	38.34 ± 1.30	35.39 ± 0.58	35.40 ± 0.59	**31.74 ± 0.97**
CLEVE	8.42 ± 0.67	8.49 ± 0.74	**8.23 ± 0.76**	8.56 ± 0.67
CRX	**14.63 ± 0.81**	17.84 ± 1.05	16.89 ± 1.00	17.34 ± 1.02
FLARE	**45.55 ± 1.06**	24332.38 ± 56.59	24332.03 ± 56.59	48.54 ± 1.21
GLASS	**13.26 ± 1.28**	11713.24 ± 72.91	11713.00 ± 72.91	15.12 ± 1.31
GLASS2	5.15 ± 0.69	4.68 ± 0.57	**4.57 ± 0.54**	5.16 ± 0.69
IRIS	**3.83 ± 0.72**	4.04 ± 0.67	3.96 ± 0.70	3.83 ± 0.72
LETTER	1349.63 ± 8.01	1385.73 ± 8.95	**1300.23 ± 8.38**	-
LIVER	16.40 ± 0.98	12.62 ± 0.79	**11.71 ± 0.65**	16.59 ± 1.01
NURSERY	**112.90 ± 2.48**	3126.39 ± 77.45	3123.62 ± 77.45	-
PRIMARY-TUMOR	**58.53 ± 1.89**	75927.03 ± 123.39	75926.94 ± 123.39	59.49 ± 1.90
SOYBEAN	**66.35 ± 1.78**	41125.59 ± 108.25	41125.46 ± 108.25	-
VOTES	**4.53 ± 0.64**	6.09 ± 0.50	6.03 ± 0.48	4.53 ± 0.65

Table 8.2: Averages and standard deviations of *LogScore* using 10% of the learning data

Dataset	SSTBMATAN	sTAN	sTAN+BMA	TBMATAN
ADULT	**16.23 ± 0.74**	16.46 ± 0.78	16.45 ± 0.83	-
AUSTRALIAN	**14.23 ± 0.85**	18.14 ± 0.91	17.74 ± 0.80	14.87 ± 1.07
BREAST	**4.32 ± 0.71**	5.26 ± 0.84	4.75 ± 0.72	5.75 ± 0.67
CAR	**7.39 ± 0.79**	8.68 ± 0.68	8.09 ± 0.58	7.55 ± 0.77
CHESS	9.70 ± 0.46	8.25 ± 0.49	8.15 ± 0.49	**7.83 ± 0.42**
CLEVE	**18.87 ± 1.08**	24.01 ± 1.31	23.57 ± 1.28	19.93 ± 1.11
CRX	**14.28 ± 0.80**	18.12 ± 0.92	17.68 ± 0.85	15.24 ± 0.90
FLARE	19.83 ± 0.67	18.55 ± 0.62	**18.54 ± 0.72**	19.96 ± 0.65
GLASS	**20.35 ± 1.01**	33.79 ± 1.14	33.86 ± 0.97	22.86 ± 1.42
GLASS2	**21.11 ± 1.35**	22.38 ± 1.53	23.40 ± 1.54	22.83 ± 1.48
IRIS	10.27 ± 1.11	8.40 ± 1.00	**8.27 ± 0.82**	10.27 ± 1.11
LETTER	**11.78 ± 0.84**	15.62 ± 0.91	15.31 ± 0.83	-
LIVER	36.39 ± 1.11	36.73 ± 1.60	**35.17 ± 1.34**	36.80 ± 1.21
NURSERY	6.82 ± 0.80	7.09 ± 0.80	**6.03 ± 0.97**	-
PRIMARY-TUMOR	**57.79 ± 1.53**	60.23 ± 1.17	59.87 ± 1.33	58.32 ± 1.55
SOYBEAN	**6.71 ± 0.59**	7.88 ± 0.71	7.80 ± 0.82	-
VOTES	6.16 ± 0.61	7.63 ± 0.93	7.76 ± 0.93	**6.03 ± 0.65**

Table 8.3: Averages and standard deviations of error rate using 50% of the learning data

Dataset	SSTBMATAN	sTAN	sTAN+BMA	TBMATAN
ADULT	**496.04 ± 3.24**	520.03 ± 3.93	518.82 ± 3.91	-
AUSTRALIAN	**11.01 ± 0.55**	14.79 ± 0.76	14.41 ± 0.59	12.22 ± 0.82
BREAST	5.97 ± 0.51	5.17 ± 0.64	**4.40 ± 0.62**	10.23 ± 0.75
CAR	16.03 ± 0.46	20.44 ± 0.51	19.73 ± 0.48	**15.87 ± 0.41**
CHESS	32.11 ± 0.49	27.32 ± 0.73	27.12 ± 0.79	**26.59 ± 0.68**
CLEVE	**6.05 ± 0.61**	7.38 ± 0.66	7.15 ± 0.63	6.53 ± 0.66
CRX	**11.51 ± 0.68**	15.62 ± 1.11	15.21 ± 1.07	12.90 ± 0.81
FLARE	**37.93 ± 0.98**	4233.42 ± 41.82	4233.31 ± 41.82	39.38 ± 1.14
GLASS	**6.50 ± 0.52**	309.52 ± 24.49	309.25 ± 24.49	9.77 ± 0.92
GLASS2	3.91 ± 0.47	3.86 ± 0.53	**3.68 ± 0.51**	4.58 ± 0.56
IRIS	1.65 ± 0.36	1.52 ± 0.37	**1.48 ± 0.34**	1.69 ± 0.36
LETTER	**416.51 ± 6.07**	574.47 ± 6.13	559.56 ± 6.17	-
LIVER	11.38 ± 0.75	10.78 ± 0.74	**10.39 ± 0.71**	12.36 ± 0.79
NURSERY	**99.30 ± 2.36**	1596.96 ± 67.06	1594.32 ± 67.06	-
PRIMARY-TUMOR	**37.30 ± 1.07**	12028.93 ± 51.79	12028.74 ± 51.79	40.90 ± 1.24
SOYBEAN	**7.01 ± 0.84**	907.34 ± 42.43	907.27 ± 42.43	-
VOTES	**3.37 ± 0.49**	5.04 ± 0.80	4.50 ± 0.69	3.39 ± 0.51

Table 8.4: Averages and standard deviations of *LogScore* using 50% of the learning data

Dataset	SSTBMATAN	sTAN	sTAN+BMA	TBMATAN
ADULT	**16.26 ± 0.71**	16.46 ± 0.68	16.42 ± 0.72	-
AUSTRALIAN	**13.74 ± 0.58**	16.49 ± 0.65	16.43 ± 0.72	13.83 ± 0.69
BREAST	**3.49 ± 0.44**	4.29 ± 0.66	3.72 ± 0.45	4.63 ± 0.58
CAR	6.03 ± 0.40	6.23 ± 0.55	6.16 ± 0.53	**5.78 ± 0.45**
CHESS	9.47 ± 0.25	7.89 ± 0.38	7.68 ± 0.44	**7.68 ± 0.22**
CLEVE	**18.22 ± 0.78**	19.99 ± 1.26	19.73 ± 1.18	18.47 ± 1.09
CRX	**13.24 ± 0.41**	15.71 ± 0.66	15.79 ± 0.74	13.47 ± 0.60
FLARE	19.91 ± 0.44	18.46 ± 0.30	**18.31 ± 0.24**	19.71 ± 0.55
GLASS	**13.80 ± 1.27**	26.58 ± 1.22	25.99 ± 1.28	18.41 ± 1.12
GLASS2	**16.43 ± 1.43**	19.61 ± 1.42	18.06 ± 1.43	19.63 ± 1.28
IRIS	**7.20 ± 1.52**	8.13 ± 1.44	**7.20 ± 1.43**	7.47 ± 1.57
LETTER	**9.71 ± 0.80**	12.69 ± 0.77	12.48 ± 0.83	-
LIVER	33.21 ± 1.07	33.36 ± 0.98	**33.19 ± 1.10**	33.99 ± 0.82
NURSERY	6.96 ± 0.78	6.62 ± 0.75	**4.81 ± 0.76**	-
PRIMARY-TUMOR	**54.08 ± 1.04**	56.74 ± 1.09	56.32 ± 0.93	54.38 ± 1.24
SOYBEAN	**5.36 ± 0.55**	5.97 ± 0.50	5.94 ± 0.49	-
VOTES	5.89 ± 0.35	6.26 ± 0.81	6.34 ± 0.56	**5.88 ± 0.74**

Table 8.5: Averages and standard deviations of error rate using 100% of the learning data

Dataset	SSTBMATAN	sTAN	sTAN+BMA	TBMATAN
ADULT	**493.81 ± 3.13**	508.10 ± 3.07	508.01 ± 3.07	-
AUSTRALIAN	**9.88 ± 0.28**	12.90 ± 0.65	12.66 ± 0.61	10.42 ± 0.43
BREAST	5.40 ± 0.46	4.85 ± 0.50	**4.28 ± 0.54**	8.52 ± 0.71
CAR	14.39 ± 0.39	16.29 ± 0.39	16.31 ± 0.41	**14.12 ± 0.40**
CHESS	31.18 ± 0.27	26.46 ± 0.46	26.22 ± 0.36	**26.05 ± 0.29**
CLEVE	**5.50 ± 0.24**	6.51 ± 0.44	6.29 ± 0.51	5.90 ± 0.31
CRX	**10.31 ± 0.49**	13.97 ± 0.68	13.76 ± 0.58	10.96 ± 0.62
FLARE	**34.12 ± 0.73**	1532.39 ± 0.62	1532.22 ± 0.65	35.56 ± 0.88
GLASS	**4.15 ± 0.76**	7.40 ± 0.59	7.12 ± 0.52	7.92 ± 1.02
GLASS2	3.11 ± 0.43	3.20 ± 0.39	**3.08 ± 0.37**	3.91 ± 0.55
IRIS	**1.12 ± 0.50**	1.18 ± 0.44	1.16 ± 0.44	1.18 ± 0.53
LETTER	**295.94 ± 4.65**	441.94 ± 5.61	433.37 ± 5.84	-
LIVER	9.97 ± 0.61	**9.59 ± 0.44**	9.72 ± 0.60	10.66 ± 0.67
NURSERY	97.67 ± 2.41	91.52 ± 2.41	**89.41 ± 2.30**	-
PRIMARY-TUMOR	**31.33 ± 0.58**	6327.87 ± 38.33	6327.64 ± 38.33	33.86 ± 0.82
SOYBEAN	4.96 ± 0.38	4.49 ± 0.51	**4.45 ± 0.48**	-
VOTES	**3.27 ± 0.40**	3.96 ± 0.55	3.76 ± 0.46	3.55 ± 0.53

Table 8.6: Averages and standard deviations of $LogScore$ using 100% of the learning data

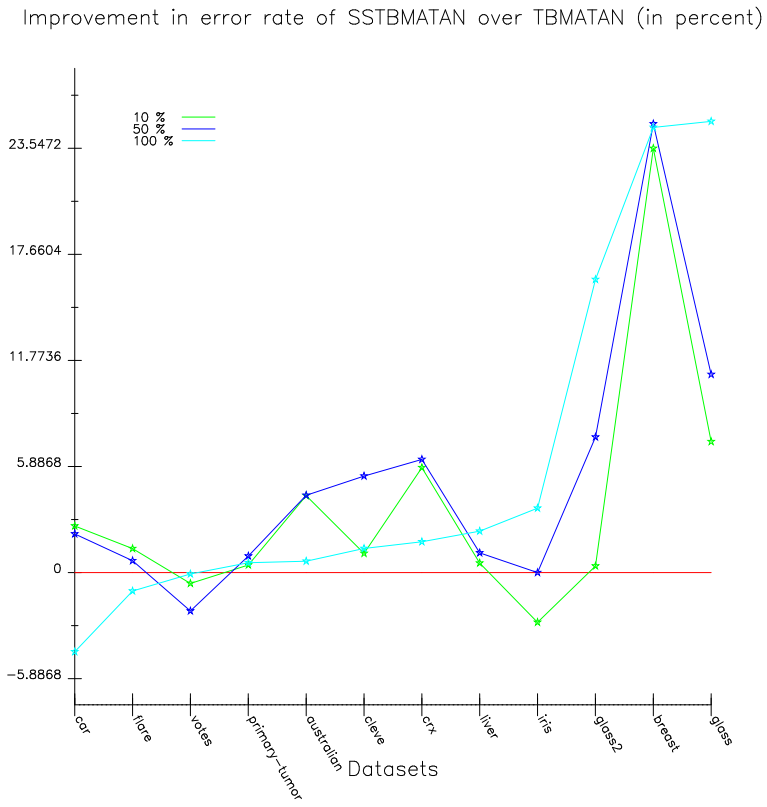


Figure 8.2: Comparison of SSTBMATAN and TBMATAN error rate

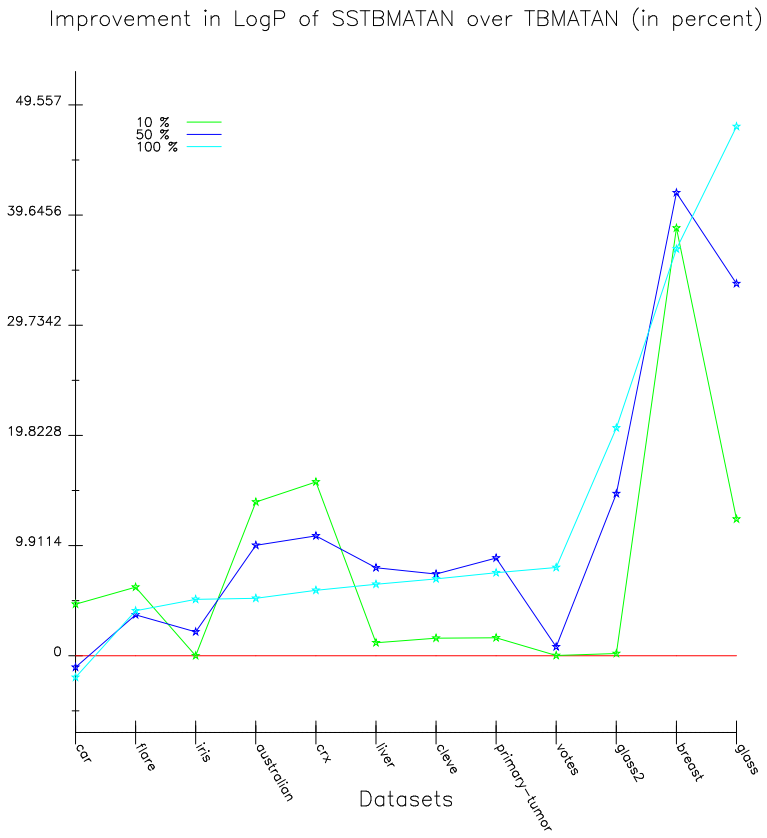


Figure 8.3: Comparison of SSTBMATAN and TBMATAN$LogScore$

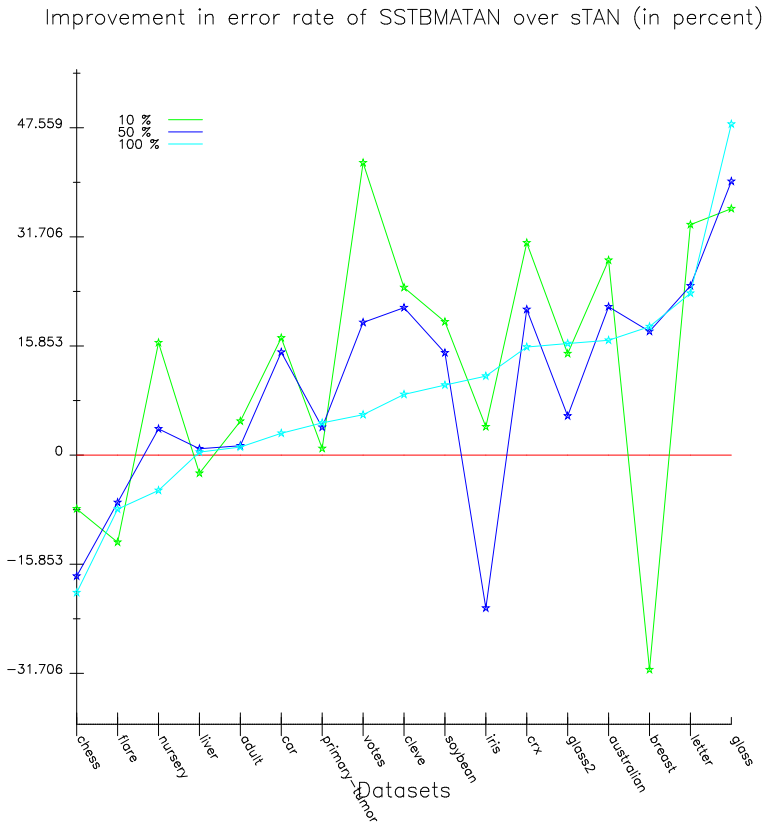


Figure 8.4: Comparison of SSTBMATAN and STAN error rate

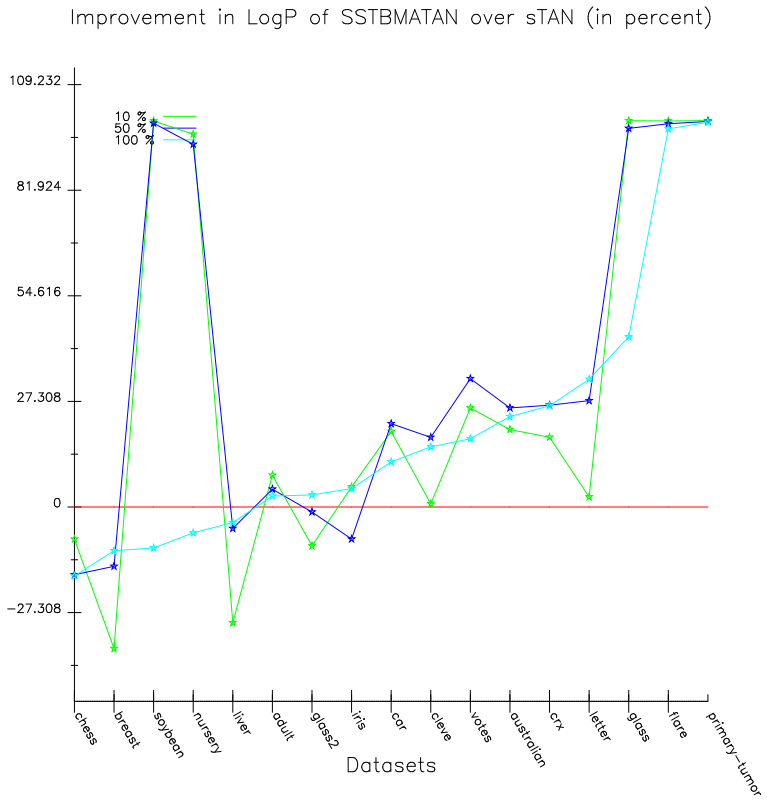


Figure 8.5: Comparison of SSTBMATAN and STAN *LogScore*

Figure 8.6: Comparison of SSTBMATAN and STAN+BMA error rate

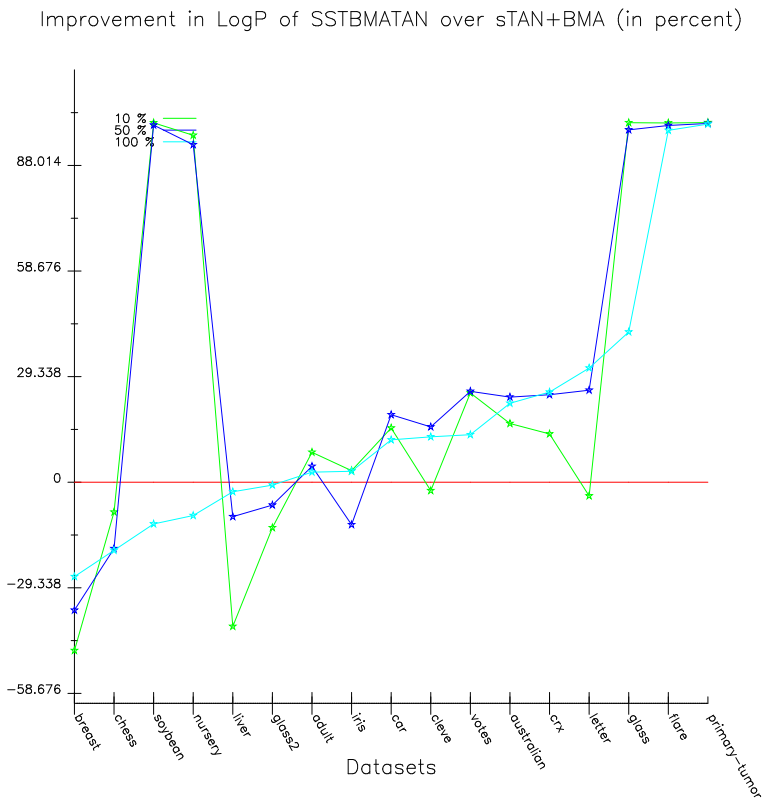


Figure 8.7: Comparison of SSTBMATAN and STAN+BMA *LogScore*

SSTBMATAN **vs** TBMATAN

The approximation to TBMATAN provided by SSTBMATAN is good for datasets such as the ones used in this experimental test. In fact, in many cases SSTBMATAN improves both error rate and *LogScore* with respect to TBMATAN, and in a statistically significant way. The improvements can be appreciated in figures 8.2 and 8.3. After performing a 5% statistical significance t-test, we have that SSTBMATAN error rate is significantly better than TBMATAN 4, 5 and 3 times with 10%, 50% and 100% of the learning data respectively, whilst TBMATAN error rate is better than SSTBMATAN in a statistically significant way only for one dataset and only with 100% of the training data. *LogScore* results favour SSTBMATAN more clearly. SSTBMATAN *LogScore* is significantly better than TBMATAN 11, 10 and 11 times with 10%, 50% and 100% of the learning data respectively, whilst TBMATAN error rate is only 0, 1 and 1 times better than SSTBMATAN in a statistically significant way. The fact that SSTBMATAN provides better error rate than TBMATAN could be explained if we understand that in the development of TBMATAN we are assuming that data is coming from a TAN model. As this is generally not the case for the datasets used in the test, probably TBMATAN is assigning too much confidence to the most probable model and SSTBMATAN, being sligthly more conservative, performs better. Testing this conjecture by means of experimental analysis based on artificial data generated from TAN distributions remains as future work.

SSTBMATAN **vs** STAN

SSTBMATAN improves both error rate and *LogScore* with respect to STAN in a statistically significant way. The improvements can be appreciated in figures 8.4 and 8.5. After performing a 5% statistical significance t-test, we have that SSTBMATAN error rate is significantly better than STAN 11, 12 and 8 times with 10%, 50% and 100% of the learning data respectively, whilst STAN error rate is only 3, 3 and 3 times better than SSTBMATAN in a statistically significant way. *LogScore* results also favour SSTBMATAN. SSTBMATAN *LogScore* is significantly better than STAN 9, 9 and 10 times with 10%, 50% and 100% of the learning data respectively, whilst STAN error rate is only 4, 3 and 4 times better than SSTBMATAN in a statistically significant way.

SSTBMATAN **vs** STAN+BMA

SSTBMATAN improves both error rate and *LogScore* with respect to STAN+BMA in a statistically significant way specially when little data is available. The improvements can be appreciated in figures 8.6 and 8.7. After performing a 5% statistical significance t-test, we have that SSTBMATAN error rate is significantly better than STAN+BMA 11, 11 and 6 times with 10%, 50% and 100% of the learning data respectively, whilst STAN+BMA error rate is only 4, 4 and 3 times better than SSTBMATAN in a statistically significant way. *LogScore* results also favour SSTBMATAN. SSTBMATAN *LogScore* is significantly better than STAN+BMA 9, 9 and 10 times with 10%, 50% and 100% of the learning data

respectively, whilst STAN+BMA error rate is only 4, 4 and 5 times better than SSTBMATAN in a statistically significant way. It is worth noting that SSTBMATAN learning time is equal to STAN learning time and much shorter than STAN+BMA learning time. On the other hand, SSTBMATAN classification time is higher.

8.5 Conclusions and Future Work

We have introduced TBMATAN a classifier based on the TAN model, decomposable distributions over TANs and Bayesian model averaging. We have seen that its implementation leads to the calculation of ill-conditioned determinants and have proposed to use an approximated implementation: SSTBMATAN.

SSTBMATAN is, to the best of our knowledge, the most accurate classifier reported with a learning time linear on the number of observations of the dataset. The accuracy increase comes at the price of increasing the classification time, making it cubic on the number of attributes. The algorithm is anytime and incremental: as long as the dataset observations are processed randomly, we can stop the learning stage anytime we need, perform some classifications and then continue learning at the only (obvious) cost of the lower accuracy of the classifications performed in the middle of the learning process. These characteristics make the algorithm very suitable for datasets with a huge number of instances.

8.5.1 Future work

If we were able to determine beforehand the impact of working with a sample in the accuracy of the predictions, in the line of Chernoff-Hoeffding bounds, we could speed up considerably the algorithm by accepting a small deviation in accuracy. In this sense, finding a result in the line of (Hulten and Domingos, 2002) remains as future work.

Being able to calculate some measure of the concentration of the posterior distribution around the TAN learned by STAN (that is, some sort of "variance") will probably allow us to determine beforehand whether TBMATAN will provide significant improvement over STAN in a dataset.

Finally, we think that all of the classifiers reviewed by Friedman et al. (Friedman et al., 1997) that are based on the Chow and Liu algorithm (Chow and Liu, 1968) can benefit from an improvement similar to the one seen here by the use of decomposable distributions and Bayesian model averaging. Formalizing the development for these classifiers and performing the empirical tests remains as future work.

Chapter 9

Maximum a Posteriori Tree Augmented Naive Bayes Classifiers

First weigh the considerations, then take the risks.

Helmuth von Moltke (1800 - 1891)

In chapter 8 we have reviewed decomposable distributions over trees and we have introduced decomposable distributions over TANs. In this chapter we present two important results for these two families of probability distributions. Concretely, we show that the problem of finding the tree (resp. TAN) with maximum a posteriori (MAP) probability is computable in poynomial time, and that the problem of finding the set of k trees (resp. TANs) with MAP probability and their relative probability weights is computable in polynomial time. In the case of trees, the main idea behind these results was presented in (Heckerman et al., 1995), prior to the introduction of decomposable distributions over trees in (Meila and Jaakkola, 2000b). To the best of our knowledge, this idea has been provided no further development and has never been related to TAN learning in the literature. In this chapter we develop the ideas sketched in (Heckerman et al., 1995) into detail and show that these results are trivially extensible to TAN models. We provide algorithms for finding the MAP and k MAP trees for decomposable distributions over trees in section 9.1. In section 9.2 we show that these results can be extended to decomposable distributions over TANs and that they can be used to create two new classifiers based on the TAN model: MAPTAN and MAPTAN+BMA. In section 9.3 we analyze the empirical performance of the classifiers introduced in section 9.2. Finally, in section 9.4 we provide some conclusions and highlight some future research lines.

9.1 Maximum a Posteriori results for decomposable distributions over trees

In this section we show that if we assume a decomposable distribution over trees as prior over the set of models, the MAP tree can be found in $\mathcal{O}((N+r^2)\cdot n^2)$ time where $r = \max_{i\in\Omega} \# X_i$. Furthermore, we also show that we can find the k MAP trees and their relative probability weights in $\mathcal{O}((N+r^2+\log(\beta(n^2,n))+k)\cdot n^2)$ time, where $\beta(m,n)$ is defined to be $\min\{i|\log^{(i)} n \leq m/n\}$ and $\log^{(i)} x$ denotes the log function iterated i times. For almost all practical considerations the time complexity of finding the k MAP trees is equivalent to $\mathcal{O}((N+r^2+k)\cdot n^2))$.

Both results are supported by the next result, that shows that computing the most probable tree under a decomposable distribution over trees with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$ can be reduced to calculating the maximum weighted spanning tree (MWST) for the graph with adjacency matrix $\log(\boldsymbol{\beta})$.

9.1.1 Calculating the most probable tree under a decomposable distribution over trees

From the definition of decomposable distribution, concretely from equations 8.2 and 8.3, it is easy to see that the most probable undirected tree given a decomposable distribution over trees with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$ comes given by

$$MPT(\boldsymbol{\beta}, \mathbf{N}') = \underset{E\in\mathcal{E}}{argmax} \prod_{u,v\in E} \beta_{u,v} \tag{9.1}$$

We can see that $MPT(\boldsymbol{\beta}, \mathbf{N}')$ does not depend on $\mathbf{N}'$. Furthermore, assuming that $\forall u, v\ ; u \neq v;\ \beta_{u,v} > 0$, we can take the logarithm of the r.h.s. having

$$MPT(\boldsymbol{\beta}, \mathbf{N}') = \underset{E\in\mathcal{E}}{argmax} \sum_{u,v\in E} \log(\beta_{u,v}) \tag{9.2}$$

Considering the matrix $\log(\boldsymbol{\beta})$ as an adjacency matrix, $MPT(\boldsymbol{\beta}, \mathbf{N}')$ is the MWST for the graph represented by that adjacency matrix. Hence, if we are given a decomposable distribution over trees with hyperparameter $\boldsymbol{\beta}$, we can find the most probable tree by calculating the logarithm of every element in the matrix and then running any algorithm for finding the MWST. The complexity of the MWST algorithm for a complete graph is $\mathcal{O}(n^2)$ (Pettie and Ramachandran, 2002).

9.1.2 Calculating the MAP tree given a prior decomposable distribution over trees

In section 8.1.2 (together with appendix B.2.2) we demonstrated that if we assume a decomposable distribution over trees with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$ as prior, the posterior distribution after a dataset $\mathcal{D}$ follows a decomposable distribution over trees with hyperparameters given by equations 8.10, 8.11 and 8.13.

```
procedure MAPTreeStructure (Dataset 𝒟,Matrix β,CountingSet N')
    var
       CountingSet N';
       Matrix lβ*;
    begin
        N'* = CalcN'Posterior(𝒟,N');
        lβ* = CalcLogBetaPosterior(β,N',N'*);
        return MWST(lβ*);

procedure CalcN'Posterior (Dataset 𝒟,CountingSet N')
    var
       CountingSet N'*;
    begin
        foreach attribute u
            foreach attribute v < u
                foreach value x_u ∈ A_u
                    foreach value x_v ∈ A_v
                        N'*_{u,v}(x_u, x_v) = N'_{u,v}(x_u, x_v);
        foreach attribute x ∈ 𝒟
            foreach attribute u
                foreach attribute v < u
                    N'*_{u,v}(x_u, x_v) = N'*_{u,v}(x_u, x_v) + 1;
        return N'*;

procedure CalcLogBetaPosterior (Matrix β,CountingSet N', N'*)
    var
       Matrix lβ*;
    begin
        foreach attribute u
            foreach attribute v < u
                lβ*_{u,v} = log β_{u,v} + CalcLogW(N',N'*,u,v);
        return lβ*;

procedure CalcLogW (CountingSet N', N'*, int u, v)
    begin
        w = 0;
        foreach value x_u ∈ A_u
            w = w + log Γ(N'_u(x_u)) - log Γ(N'*_u(x_u));
        foreach value x_v ∈ A_v
            w = w + log Γ(N'_v(x_v)) - log Γ(N'*_v(x_v));
        foreach value x_u ∈ A_u
            foreach value x_v ∈ A_v
                w = w + log Γ(N'*_{u,v}(x_u, x_v)) - log Γ(N'_{u,v}(x_u, x_v));
        return w;
```

Algorithm 9: Computation of the MAP tree

Since the posterior is a decomposable distribution over trees, we can apply the former result for finding the most probable tree over it and we will get the MAP tree. We can translate this result into algorithm 9, that calculates the MAP tree given a dataset $\mathcal{D}$ and prior hyperparameters $\boldsymbol{\beta}$,$\mathbf{N}'$. Since the computation of MWST is $\mathcal{O}(n^2)$, the time complexity of `MAPTreeStructure` is bounded by `CalcN'Posterior`, which has complexity $\mathcal{O}((N + r^2) \cdot n^2)$.

9.1.3 Calculating the k MAP trees and their relative weights given a prior decomposable distribution over trees

The problem of computing the k MWST in order is well known and can be solved in $\mathcal{O}((\log(\beta(n^2, n)) + k) \cdot n^2)$ for a complete graph (Katoh et al., 1981). It is easy to see that if in the last step of `MAPTreeStructure` instead of calculating the MWST we calculate the k MWST and their relative weights as shown in algorithm 10, the algorithm will return the k MAP trees and their relative probabilities. The time complexity of the new algorithm is simply the addition of the complexity of `CalcN'Posterior` with that of computing the k MAP trees and that of computing the weights, giving $\mathcal{O}((N + r^2 + \log(\beta(n^2, n)) + k) \cdot n^2)$ which, as previouly mentioned, can be understood as $\mathcal{O}((N + r^2 + k) \cdot n^2)$ for all practical purposes.

```
procedure k-MAPTrees (Dataset D,Matrix β,CountingSet N', int k)
    var
       CountingSet N';
       WeightedTreeSet WTS;
       Matrix lβ*;
    begin
        N'* = CalcN'Posterior(D,N');
        lβ* = CalcLogBetaPosterior(β,N',N'*);
        WTS = k-MWST(lβ*,k);
        CalcTreeWeights(WTS,lβ*);
        return WTS;

procedure CalcTreeWeights (WeightedTreeSet WTS, Matrix lβ*)
    begin
        foreach tree T ∈ WTS
            w = 0;
            foreach edge (u,v) ∈ T
                w = w + lβ*_{u,v};
            fixWeight(T,exp(w));
        normalizeWeightsToSumOne(WTS);
```

Algorithm 10: Computation of the k MAP trees

9.2 MAPTAN and MAPTAN+BMA classifiers

In the previous section we have seen that it is possible to efficiently calculate the MAP tree and the k MAP trees under a decomposable distribution over trees. In this section we show that these results can be trivially extended to decomposable distributions over TANs and provide the corresponding algorithms. After that, we show that two new classifiers, MAPTAN and MAPTAN+BMA, can be created having these results as basis. Finally we briefly discuss the characteristics and benefits of these two new classifiers.

9.2.1 Maximum a Posteriori results for decomposable distributions over TANs

It is easy to see that results parallel to the ones presented in section 9.1 hold for decomposable distributions over TANs. Concretely, algorithm 11 computes the undirected tree structure underlying the MAP TAN in $\mathcal{O}((N + r^3) \cdot n^2)$ where $r = \max(\max_{i \in V} \#A_i, \#C)$. Equivalently, algorithm 12 calculates the k undirected tree structures underlying the k MAP TAN models and their relative weights in $\mathcal{O}((N+r^3+\log(\beta(n^2,n))+k)\cdot n^2)$ which can be understood as $\mathcal{O}((N+r^3+k)\cdot n^2))$ for all practical purposes.

9.2.2 Constructing the MAPTAN and MAPTAN+BMA classifiers

The previosly introduced results allow us to efficiently compute MAP TAN structures. We know from equation B.58 that

$$P(\mathcal{C} = s_C | \mathcal{V} = S, E, \xi) \propto h_0^{S,S_C} \prod_{u,v \in \overline{E}} h_{u,v}^{S,S_C} \tag{9.3}$$

In fact, given a undirected TAN structure E, it is easy to see that the probability distribution $P(\mathcal{C} = s_C | \mathcal{V} = S, E, \xi)$ can be represented as a TAN model with structure $\overline{E}^*$, such that its undirected version coincides with E and a parameter set given by

$$\begin{aligned} \theta_{u|v,C}(s_u, s_v, s_C) &= \frac{N'_{u,v,C}(s_u, s_v, s_C)}{N'_{v,C}(s_v, s_C)} \\ \theta_{u|C}(s_u, s_C) &= \frac{N'_{u,C}(s_u, s_C)}{N'_C(s_C)} \\ \theta_C(s_C) &= \frac{N'_C(s_C)}{N'} \end{aligned} \tag{9.4}$$

A similar result in the case of decomposable distribution over trees can also be found in (Meila and Jordan, 2000).

Given a decomposable prior we can calculate the decomposable posterior using the result in section 8.2.3 and then apply the result we have just enunciated to the posterior. The posterior probability distribution $P(\mathcal{C} = s_C | \mathcal{V} = S, E, \mathcal{D}, \xi)$ can be represented as a TAN model with structure $\overline{E}^*$, such that its undirected version coincides with E and its parameter set is given by

```
procedure MAPTANStructure (Dataset 𝒟,Matrix β,CountingSet N′)
    var
       CountingSet N′;
       Matrix lβ*;
    begin
        N′* = CalcN'PosteriorTAN(𝒟,N′);
        lβ* = CalcLogBetaPosteriorTAN(β,N′,N′*);
        return MWST(lβ*);

procedure CalcN'PosteriorTAN (Dataset 𝒟,CountingSet N′)
    var
       CountingSet N′*;
    begin
        foreach attribute u
            foreach attribute v < u
                foreach value x_u ∈ A_u
                    foreach value x_v ∈ A_v
                        foreach value c ∈ C
                            N′*_{u,v,C}(x_u, x_v, c) = N′_{u,v}(x_u, x_v, c);
        foreach attribute x ∈ 𝒟
            foreach attribute u
                foreach attribute v < u
                    N′*_{u,v,C}(x_u, x_v, x_C) = N′*_{u,v,C}(x_u, x_v, x_C) + 1;
        return N′*;

procedure CalcLogBetaPosteriorTAN (Matrix β,CountingSet N′, N′*)
    var
       Matrix lβ*;
    begin
        foreach attribute u
            foreach attribute v < u
                lβ*_{u,v} = log β_{u,v} + CalcLogWTAN(N′,N′*,u,v);
        return lβ*;

procedure CalcLogWTAN (CountingSet N′, N′*, int u, v)
    begin
        w = 0;
        foreach value c ∈ C
            foreach value x_u ∈ A_u
                w = w + logΓ(N′_{u,C}(x_u, c)) - logΓ(N′*_{u,C}(x_u, c));
            foreach value x_v ∈ A_v
                w = w + logΓ(N′_{v,C}(x_v, c)) - logΓ(N′*_{v,C}(x_v, c));
            foreach value x_u ∈ A_u
                foreach value x_v ∈ A_v
                    w = w + logΓ(N′*_{u,v,C}(x_u, x_v, c)) - logΓ(N′_{u,v,C}(x_u, x_v, c));
        return w;
```

Algorithm 11: Computation of the MAP TAN

$$\begin{aligned}\theta_{u|v,C}(s_u, s_v, s_C) &= \frac{N'_{u,v,C}(s_u,s_v,s_C)+N_{u,v,C}(s_u,s_v,s_C)}{N'_{v,C}(s_v,s_C)+N_{v,C}(s_v,s_C)} \\ \theta_{u|C}(s_u, s_C) &= \frac{N'_{u,C}(s_u,s_C)+N_{u,C}(s_u,s_C)}{N'_{C}(s_C)+N_{C}(s_C)} \\ \theta_{C}(s_C) &= \frac{N'_{C}(s_C)+N_{C}(s_C)}{N'+N}\end{aligned} \tag{9.5}$$

Given this result and our previous results for determining the MAP TAN structure and the k MAP TAN structures and their relative probability weights, it is very easy to construct two new classifiers by simple composition. First of all we have to fix a set of prior hyperparameters. In section 8.2.4 we argued that

$$\forall u, v \ ; \ 1 \leq u \neq v \ \leq n \ ; \ \beta_{u,v} = 1 \tag{9.6}$$

$$\forall u, v; 1 \leq u \neq v \leq n; \forall j \in A_v; \forall i \in A_u; \forall c \in C; N'_{v,u,C}(j, i, c) = \frac{\lambda}{\#C\#A_u\#A_v} \tag{9.7}$$

where λ is an equivalent sample size, provide a reasonable choice of the hyperparameters if no information from the domain is available.

MAPTAN classifier

After fixing the prior hyperparameters, the learning step for MAPTAN classifier consist in:

1. Applying algorithm 11 to find the undirected dependence tree E underlying the MAP TAN structure given a dataset $\mathcal{D}$.
2. Randomly choose a root, create a directed tree $\overline{E}$ and from it a directed TAN structure $\overline{E}^*$.
3. Use equation 9.5 to fix the TAN parameters.

For classifying an unclassified observation, we have to apply the TAN that has been learned for each of the $\#C$ classes to construct a probability distribution over the values of the class C and then choose the most probable class.

This classification algorithm runs in $\mathcal{O}((N+r^3)\cdot n^2)$ learning time and $\mathcal{O}(nr)$ classification time.

MAPTAN+BMA classifier

After fixing the prior hyperparameters, the learning stage for MAPTAN+BMA classifier consists in:

1. Applying algorithm 12 to find the k undirected trees underlying the k MAP TAN structures and their relative probability weights given a dataset $\mathcal{D}$.
2. Generate a TAN model for each of the undirected tree structures as we did in MAPTAN.

```
procedure k-MAPTANs (Dataset 𝒟,Matrix β,CountingSet N', int k)
    var
       CountingSet N';
       WeightedTreeSet WTS;
       Matrix lβ*;
    begin
        N'* = CalcN'PosteriorTAN(𝒟,N');
        lβ* = CalcLogBetaPosteriorTAN(β,N',N'*);
        WTS = k-MWST(lβ*,k);
        CalcTreeWeights(WTS,lβ*);
        return WTS;
```

Algorithm 12: Computation of the k MAP TANs

3. Assign to each TAN model the weight of its corresponding undirected tree using equation 8.23.

The resulting probabilistic model will be a mixture of TANs. For classifying an unclassified observation, we have to apply the k TAN models for the $\#C$ classes and calculate the weighted average to construct a probability distribution over the values of the class C and then choose the most probable class.

This classification algorithm runs in $\mathcal{O}((N + r^3 + \log(\beta(n^2, n)) + k) \cdot n^2)$ learning time and $\mathcal{O}(nrk)$ classification time.

Relevant characteristics of MAPTAN and MAPTAN+BMA

We have shown that decomposable distributions over TANs can be used to construct two well founded classifiers: MAPTAN and MAPTAN+BMA. These classifiers have some characteristics worth discussing. In the introduction to chapter 7, we highlighted two possible ways in which the TAN classifier, as presented in (Friedman et al., 1997), could be improved: by taking into account model uncertainty and by providing a theoretically well founded explanation for the use of softening.

The first weak point is that the classifier does not take into account model uncertainty. In chapter 7 we proposed STAN+BMA, based on empirical local Bayesian model averaging as a possible fix for this weak point. We showed that STAN+BMA improved STAN accuracy at the cost of increasing considerably the learning time complexity. In chapter 8 we used the fact that decomposable distributions over TANs allow the tractable calculation of the model averaging integral to construct TBMATAN, a classifier that takes into account model uncertainty in a theoretically well founded way. After its implementation, we realized that the computation of TBMATAN leads to the calculation of an ill-conditioned determinant and proposed SSTBMATAN, an approximation of TBMATAN that showed better accuracy results. In spite of that, while SSTBMATAN learning time complexity was equivalent to that of STAN, its classification time complexity was

$\mathcal{O}(n^3 r)$, clearly higher than STAN's $\mathcal{O}(nr)$. MAPTAN+BMA can be seen as sharing some of the good properties of STAN, STAN+BMA, TBMATAN and SSTBMATAN. As TBMATAN, it provides a theoretically well founded way of dealing with model uncertainty. Its learning time complexity regarding N is almost equivalent to that of STAN and it grows polynomially on k. It has a classification time complexity, $\mathcal{O}(nrk)$ as good as that of STAN+BMA, and reasonably higher than that of STAN. Furtermore, as with STAN+BMA, we can use k as an *effort knob*, in the sense of (Thearling, 1998), hence providing a useful feature for data mining users that allows them to decide how much computational power they want to spend in the task. In our opinion, MAPTAN+BMA provides the better complexity tradeoff of the four classifiers designed in this thesis to deal with model uncertainty when learning TAN.

The second improvement consists in providing a theoretically well founded explanation for the use of softening. Both MAPTAN and MAPTAN+BMA can be interpreted as using softening in both the structure search and the parameter fixing. This softening appears, in a natural way, as the result of assuming a decomposable distribution over TANs as the prior over the set of models. In our opinion MAPTAN is theoretically more appealing than STAN.

Finally, both MAPTAN and MAPTAN+BMA, share with TBMATAN and SSTBMATAN the relevant characteristic of allowing the use of some form of prior information if such is available, specially structure related information. For example, if we have expert knowledge that tell us that one of the edges of the tree is much more (equiv. much less) likely than the others it is very easy to incorporate this knowledge when fixing the prior hyperparameter matrix $\boldsymbol{\beta}$. Evidently, as was pointed out in (Meila and Jaakkola, 2000b), decomposable distributions do not allow the expression of some types of prior information such as "if edge (u, v) exists then edge (w, z) is very likely to exist".

9.3 Empirical results

We tested four algorithms over 17 datasets from the Irvine repository (Blake et al., 1998). The dataset characteristics are described in section 6.4.1.

To discretize continuous attributes we used equal frequency discretization with 5 intervals. For each dataset and algorithm we tested both error rate and *LogScore* (see section 6.4). For the evaluation of both error rate and *LogScore* we used 10 fold cross validation. We tested the algorithm with the 10%, 50% and 100% of the learning data for each fold, in order to get an idea of the influence of the amount of data in the behaviors of both error rate and *LogScore* for the algorithm.

The error rates appear in Tables 9.1,9.3,9.5 with the best method for each dataset boldfaced. *LogScore*'s appear in Tables 9.2,9.4,9.6. The columns of the tables are the induction methods and the rows are the datasets. The meaning of the column headers are:

- SSTBMATAN is the method described in section 8.3.

- STAN is the softened TAN induction algorithm as presented in (Friedman et al., 1997).
- STAN+BMA is the TAN induction algorithm using ELBMA to deal with model uncertainty as presented in section 7.1.2.
- MAPTAN, is the classifier based on the MAP TAN model described in section 9.2.2.
- MAPTAN+BMA is the classifier based on the weighted average of the k MAP TAN models described also in section 9.2.2.

9.3.1 Interpretation of the results

Summarizing the empirical results in the tables, we can conclude that:

- MAPTAN improves STAN error rate for most datasets and has a similar *LogScore*.
- MAPTAN+BMA improves MAPTAN's *LogScore* for most datasets. When little data is available, it also improves its error rate.
- MAPTAN+BMA improves STAN+BMA error rate and *LogScore* for many datasets.
- SSTBMATAN improves MAPTAN+BMA error rate and *LogScore* specially when little data is available.

In the rest of the section we discuss and justify these assertions into more detail.

MAPTAN **vs** STAN

MAPTAN improves STAN error rate in a statistically significant way for most datasets and has a similar *LogScore*. This can be appreciated in figures 9.1 and 9.2. After performing a 5% statistical significance t-test, we have that MAPTAN error rate is significantly better than STAN for 12, 11 and 7 datasets with 10%, 50% and 100% of the learning data respectively, whilst STAN error rate is better than MAPTAN in a statistically significant way only for 4, 2 and 2 datasets. *LogScore* results favor MAPTAN slightly. MAPTAN *LogScore* is significantly better than STAN for 6, 9 and 9 datasets with 10%, 50% and 100% of the learning data respectively, whilst STAN *LogScore* is better than MAPTAN in a statistically significant way for 7, 5 and 6 datasets respectively.

Dataset	MAPTAN	MAPTAN+BMA	SSTBMATAN	sTAN	sTAN+BMA
ADULT	17.18 ± 0.68	17.19 ± 0.71	**16.73 ± 0.74**	17.60 ± 0.82	17.60 ± 0.80
AUSTRALIAN	19.91 ± 1.14	19.62 ± 1.13	**18.20 ± 1.11**	25.39 ± 1.18	24.96 ± 1.13
BREAST	17.23 ± 1.21	16.89 ± 1.28	11.45 ± 1.22	8.73 ± 0.87	**7.73 ± 0.93**
CAR	17.19 ± 1.04	16.50 ± 0.84	**16.08 ± 0.85**	19.38 ± 0.95	17.60 ± 0.77
CHESS	9.55 ± 0.80	**9.48 ± 0.86**	11.74 ± 0.89	10.89 ± 0.56	10.91 ± 0.53
CLEVE	28.12 ± 1.68	28.14 ± 1.59	**24.49 ± 1.12**	32.37 ± 1.00	31.89 ± 1.27
CRX	19.77 ± 0.91	19.16 ± 1.00	**17.39 ± 0.98**	25.14 ± 0.87	24.18 ± 0.98
FLARE	23.50 ± 1.09	23.16 ± 1.09	22.46 ± 0.94	19.94 ± 0.85	**19.92 ± 0.88**
GLASS	47.02 ± 1.66	45.72 ± 1.59	**37.99 ± 1.52**	59.19 ± 1.78	58.54 ± 1.83
GLASS2	33.69 ± 1.74	32.87 ± 1.82	**32.18 ± 2.18**	37.75 ± 1.39	36.63 ± 1.37
IRIS	28.67 ± 2.33	26.27 ± 2.30	**24.80 ± 2.13**	25.87 ± 3.07	**24.80 ± 2.96**
LETTER	30.22 ± 0.96	30.19 ± 0.97	**24.02 ± 0.93**	36.11 ± 1.39	34.68 ± 1.37
LIVER	45.52 ± 1.26	44.96 ± 1.06	43.51 ± 1.14	42.39 ± 0.94	**41.24 ± 1.37**
NURSERY	7.87 ± 1.03	7.57 ± 1.04	**7.43 ± 0.98**	8.88 ± 1.12	8.50 ± 1.12
PRIMARY-TUMOR	74.52 ± 1.73	74.28 ± 1.66	**70.98 ± 1.77**	71.67 ± 1.54	71.73 ± 1.44
SOYBEAN	26.53 ± 1.30	26.51 ± 1.33	**24.83 ± 1.14**	30.79 ± 1.28	30.82 ± 1.33
VOTES	9.61 ± 0.94	9.67 ± 0.99	**8.13 ± 0.59**	14.14 ± 0.93	14.13 ± 0.71

Table 9.1: Averages and standard deviations of error rate using 10% of the learning data

Dataset	MAPTAN	MAPTAN+BMA	SSTBMATAN	sTAN	sTAN+BMA
ADULT	562.25 ± 3.75	561.39 ± 3.71	**520.49 ± 3.40**	567.09 ± 3.92	567.64 ± 4.00
AUSTRALIAN	18.54 ± 0.95	17.68 ± 0.96	**14.28 ± 0.76**	17.85 ± 0.64	17.06 ± 0.60
BREAST	23.59 ± 1.67	18.24 ± 1.56	11.09 ± 1.36	8.12 ± 0.69	**7.56 ± 0.65**
CAR	34.89 ± 1.02	32.79 ± 0.98	**31.02 ± 0.93**	38.55 ± 0.91	36.52 ± 0.86
CHESS	32.50 ± 0.89	**32.25 ± 0.91**	38.34 ± 1.30	35.39 ± 0.58	35.40 ± 0.59
CLEVE	11.15 ± 1.06	10.09 ± 0.96	8.42 ± 0.67	8.49 ± 0.74	**8.23 ± 0.76**
CRX	19.44 ± 1.06	18.30 ± 1.00	**14.63 ± 0.81**	17.84 ± 1.05	16.89 ± 1.00
FLARE	51.12 ± 1.17	49.48 ± 1.15	**45.55 ± 1.06**	24332.38 ± 56.59	24332.03 ± 56.59
GLASS	20.49 ± 1.45	17.14 ± 1.40	**13.26 ± 1.28**	11713.24 ± 72.91	11713.00 ± 72.91
GLASS2	6.45 ± 0.79	5.49 ± 0.64	5.15 ± 0.69	4.68 ± 0.57	**4.57 ± 0.54**
IRIS	4.58 ± 0.68	4.06 ± 0.69	**3.83 ± 0.72**	4.04 ± 0.67	3.96 ± 0.70
LETTER	3535.93 ± 12.92	3495.14 ± 13.52	1349.63 ± 8.01	1385.73 ± 8.95	**1300.23 ± 8.38**
LIVER	18.71 ± 0.95	15.87 ± 0.92	16.40 ± 0.98	12.62 ± 0.79	**11.71 ± 0.65**
NURSERY	112.72 ± 2.47	**111.95 ± 2.47**	112.90 ± 2.48	3126.39 ± 77.45	3123.62 ± 77.45
PRIMARY-TUMOR	71.74 ± 2.08	69.08 ± 2.05	**58.53 ± 1.89**	75927.03 ± 123.39	75926.94 ± 123.39
SOYBEAN	68.52 ± 1.77	**65.29 ± 1.55**	66.35 ± 1.78	41125.59 ± 108.25	41125.46 ± 108.25
VOTES	5.66 ± 0.66	5.17 ± 0.60	**4.53 ± 0.64**	6.09 ± 0.50	6.03 ± 0.48

Table 9.2: Averages and standard deviations of *LogScore* using 10% of the learning data

Dataset	MAPTAN	MAPTAN+BMA	SSTBMATAN	sTAN	sTAN+BMA
ADULT	16.26 ± 0.75	16.28 ± 0.77	**16.23 ± 0.74**	16.46 ± 0.78	16.45 ± 0.83
AUSTRALIAN	15.36 ± 0.94	15.13 ± 1.09	**14.23 ± 0.85**	18.14 ± 0.91	17.74 ± 0.80
BREAST	5.92 ± 0.74	5.84 ± 0.78	**4.32 ± 0.71**	5.26 ± 0.84	4.75 ± 0.72
CAR	7.62 ± 0.75	7.55 ± 0.76	**7.39 ± 0.79**	8.68 ± 0.68	8.09 ± 0.58
CHESS	**7.87 ± 0.44**	7.90 ± 0.44	9.70 ± 0.46	8.25 ± 0.49	8.15 ± 0.49
CLEVE	19.82 ± 1.30	20.27 ± 1.27	**18.87 ± 1.08**	24.01 ± 1.31	23.57 ± 1.28
CRX	15.47 ± 1.01	15.30 ± 1.02	**14.28 ± 0.80**	18.12 ± 0.92	17.68 ± 0.85
FLARE	19.83 ± 0.72	19.81 ± 0.65	19.83 ± 0.67	18.55 ± 0.62	**18.54 ± 0.72**
GLASS	24.02 ± 1.22	23.31 ± 1.48	**20.35 ± 1.01**	33.79 ± 1.14	33.86 ± 0.97
GLASS2	23.69 ± 1.62	22.81 ± 1.61	**21.11 ± 1.35**	22.38 ± 1.53	23.40 ± 1.54
IRIS	11.60 ± 1.22	11.07 ± 1.08	10.27 ± 1.11	8.40 ± 1.00	**8.27 ± 0.82**
LETTER	14.79 ± 0.78	14.79 ± 0.78	**11.78 ± 0.84**	15.62 ± 0.91	15.31 ± 0.83
LIVER	37.33 ± 1.16	36.90 ± 1.15	36.39 ± 1.11	36.73 ± 1.60	**35.17 ± 1.34**
NURSERY	6.39 ± 0.92	6.37 ± 0.89	6.82 ± 0.80	7.09 ± 0.80	**6.03 ± 0.97**
PRIMARY-TUMOR	59.15 ± 1.67	59.09 ± 1.63	**57.79 ± 1.53**	60.23 ± 1.17	59.87 ± 1.33
SOYBEAN	6.64 ± 0.77	**6.50 ± 0.85**	6.71 ± 0.59	7.88 ± 0.71	7.80 ± 0.82
VOTES	6.22 ± 0.83	6.25 ± 0.84	**6.16 ± 0.61**	7.63 ± 0.93	7.76 ± 0.93

Table 9.3: Averages and standard deviations of error rate using 50% of the learning data

Dataset	MAPTAN	MAPTAN+BMA	SSTBMATAN	sTAN	sTAN+BMA
ADULT	507.82 ± 3.82	507.52 ± 3.81	**496.04 ± 3.24**	520.03 ± 3.93	518.82 ± 3.91
AUSTRALIAN	12.86 ± 0.82	12.57 ± 0.84	**11.01 ± 0.55**	14.79 ± 0.76	14.41 ± 0.59
BREAST	10.95 ± 0.67	9.20 ± 0.69	5.97 ± 0.51	5.17 ± 0.64	**4.40 ± 0.62**
CAR	15.96 ± 0.44	**15.90 ± 0.40**	16.03 ± 0.46	20.44 ± 0.51	19.73 ± 0.48
CHESS	26.70 ± 0.66	**26.66 ± 0.68**	32.11 ± 0.49	27.32 ± 0.73	27.12 ± 0.79
CLEVE	6.83 ± 0.69	6.70 ± 0.66	**6.05 ± 0.61**	7.38 ± 0.66	7.15 ± 0.63
CRX	13.28 ± 0.89	12.93 ± 0.85	**11.51 ± 0.68**	15.62 ± 1.11	15.21 ± 1.07
FLARE	39.81 ± 1.16	39.45 ± 1.15	**37.93 ± 0.98**	4233.42 ± 41.82	4233.31 ± 41.82
GLASS	11.05 ± 0.73	9.37 ± 0.84	**6.50 ± 0.52**	309.52 ± 24.49	309.25 ± 24.49
GLASS2	5.06 ± 0.73	4.64 ± 0.64	3.91 ± 0.47	3.86 ± 0.53	**3.68 ± 0.51**
IRIS	1.87 ± 0.35	1.77 ± 0.34	1.65 ± 0.36	1.52 ± 0.37	**1.48 ± 0.34**
LETTER	1030.65 ± 9.50	1030.65 ± 9.50	**416.51 ± 6.07**	574.47 ± 6.13	559.56 ± 6.17
LIVER	13.03 ± 0.89	12.21 ± 0.77	11.38 ± 0.75	10.78 ± 0.74	**10.39 ± 0.71**
NURSERY	96.60 ± 2.40	**96.52 ± 2.42**	99.30 ± 2.36	1596.96 ± 67.06	1594.32 ± 67.06
PRIMARY-TUMOR	44.24 ± 1.25	43.00 ± 1.23	**37.30 ± 1.07**	12028.93 ± 51.79	12028.74 ± 51.79
SOYBEAN	6.79 ± 0.83	**6.47 ± 0.74**	7.01 ± 0.84	907.34 ± 42.43	907.27 ± 42.43
VOTES	3.66 ± 0.58	3.54 ± 0.52	**3.37 ± 0.49**	5.04 ± 0.80	4.50 ± 0.69

Table 9.4: Averages and standard deviations of *LogScore* using 50% of the learning data

Dataset	MAPTAN	MAPTAN+BMA	SSTBMATAN	sTAN	sTAN+BMA
ADULT	16.35 ± 0.73	16.35 ± 0.73	**16.26 ± 0.71**	16.46 ± 0.68	16.42 ± 0.72
AUSTRALIAN	13.68 ± 0.75	**13.65 ± 0.74**	13.74 ± 0.58	16.49 ± 0.65	16.43 ± 0.72
BREAST	4.75 ± 0.53	4.63 ± 0.48	**3.49 ± 0.44**	4.29 ± 0.66	3.72 ± 0.45
CAR	**5.76 ± 0.52**	5.78 ± 0.45	6.03 ± 0.40	6.23 ± 0.55	6.16 ± 0.53
CHESS	7.71 ± 0.25	**7.67 ± 0.21**	9.47 ± 0.25	7.89 ± 0.38	7.68 ± 0.44
CLEVE	18.74 ± 1.15	18.53 ± 1.18	**18.22 ± 0.78**	19.99 ± 1.26	19.73 ± 1.18
CRX	13.67 ± 0.53	13.53 ± 0.58	**13.24 ± 0.41**	15.71 ± 0.66	15.79 ± 0.74
FLARE	19.71 ± 0.49	19.71 ± 0.55	19.91 ± 0.44	18.46 ± 0.30	**18.31 ± 0.24**
GLASS	18.46 ± 1.20	18.74 ± 1.29	**13.80 ± 1.27**	26.58 ± 1.22	25.99 ± 1.28
GLASS2	19.81 ± 0.85	20.25 ± 1.39	**16.43 ± 1.43**	19.61 ± 1.42	18.06 ± 1.43
IRIS	7.73 ± 1.70	7.47 ± 1.66	**7.20 ± 1.52**	8.13 ± 1.44	**7.20 ± 1.43**
LETTER	11.49 ± 0.74	11.49 ± 0.74	**9.71 ± 0.80**	12.69 ± 0.77	12.48 ± 0.83
LIVER	34.35 ± 0.86	33.99 ± 0.77	33.21 ± 1.07	33.36 ± 0.98	**33.19 ± 1.10**
NURSERY	6.33 ± 0.89	6.26 ± 0.91	6.96 ± 0.78	6.62 ± 0.75	**4.81 ± 0.76**
PRIMARY-TUMOR	55.09 ± 1.24	54.68 ± 1.02	**54.08 ± 1.04**	56.74 ± 1.09	56.32 ± 0.93
SOYBEAN	5.47 ± 0.62	**5.27 ± 0.62**	5.36 ± 0.55	5.97 ± 0.50	5.94 ± 0.49
VOTES	**5.89 ± 0.74**	5.89 ± 0.72	5.89 ± 0.35	6.26 ± 0.81	6.34 ± 0.56

Table 9.5: Averages and standard deviations of error rate using 100% of the learning data

Dataset	MAPTAN	MAPTAN+BMA	SSTBMATAN	sTAN	sTAN+BMA
ADULT	495.88 ± 3.68	495.70 ± 3.67	**493.81 ± 3.13**	508.10 ± 3.07	508.01 ± 3.07
AUSTRALIAN	10.65 ± 0.46	10.47 ± 0.44	**9.88 ± 0.28**	12.90 ± 0.65	12.66 ± 0.61
BREAST	8.96 ± 0.87	7.89 ± 0.61	5.40 ± 0.46	4.85 ± 0.50	**4.28 ± 0.54**
CAR	**14.11 ± 0.40**	14.12 ± 0.40	14.39 ± 0.39	16.29 ± 0.39	16.31 ± 0.41
CHESS	26.12 ± 0.40	**26.09 ± 0.32**	31.18 ± 0.27	26.46 ± 0.46	26.22 ± 0.36
CLEVE	6.10 ± 0.43	6.05 ± 0.38	**5.50 ± 0.24**	6.51 ± 0.44	6.29 ± 0.51
CRX	11.34 ± 0.62	11.05 ± 0.60	**10.31 ± 0.49**	13.97 ± 0.68	13.76 ± 0.58
FLARE	35.82 ± 0.92	35.61 ± 0.90	**34.12 ± 0.73**	1532.39 ± 0.62	1532.22 ± 0.65
GLASS	8.53 ± 1.05	7.50 ± 1.03	**4.15 ± 0.76**	7.40 ± 0.59	7.12 ± 0.52
GLASS2	4.20 ± 0.56	3.91 ± 0.54	3.11 ± 0.43	3.20 ± 0.39	**3.08 ± 0.37**
IRIS	1.29 ± 0.53	1.22 ± 0.53	**1.12 ± 0.50**	1.18 ± 0.44	1.16 ± 0.44
LETTER	612.99 ± 7.71	612.99 ± 7.71	**295.94 ± 4.65**	441.94 ± 5.61	433.37 ± 5.84
LIVER	10.79 ± 0.63	10.61 ± 0.66	9.97 ± 0.61	**9.59 ± 0.44**	9.72 ± 0.60
NURSERY	94.59 ± 2.45	94.57 ± 2.43	97.67 ± 2.41	91.52 ± 2.41	**89.41 ± 2.30**
PRIMARY-TUMOR	35.28 ± 0.99	34.64 ± 0.94	**31.33 ± 0.58**	6327.87 ± 38.33	6327.64 ± 38.33
SOYBEAN	3.45 ± 0.50	**3.38 ± 0.49**	4.96 ± 0.38	4.49 ± 0.51	4.45 ± 0.48
VOTES	3.74 ± 0.59	3.57 ± 0.58	**3.27 ± 0.40**	3.96 ± 0.55	3.76 ± 0.46

Table 9.6: Averages and standard deviations of *LogScore* using 100% of the learning data

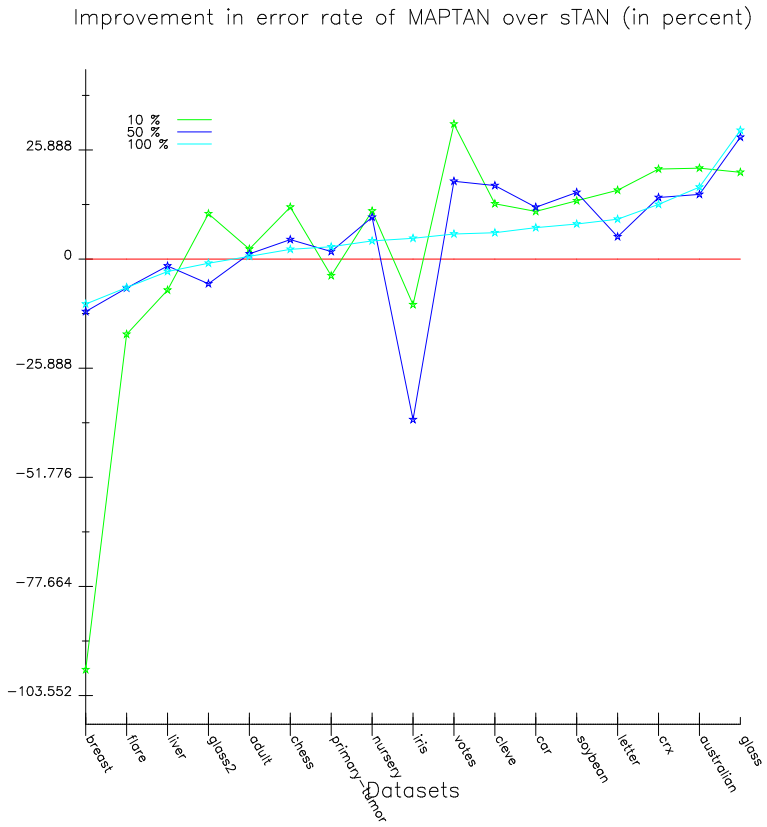


Figure 9.1: Comparison of MAPTAN and STAN error rate

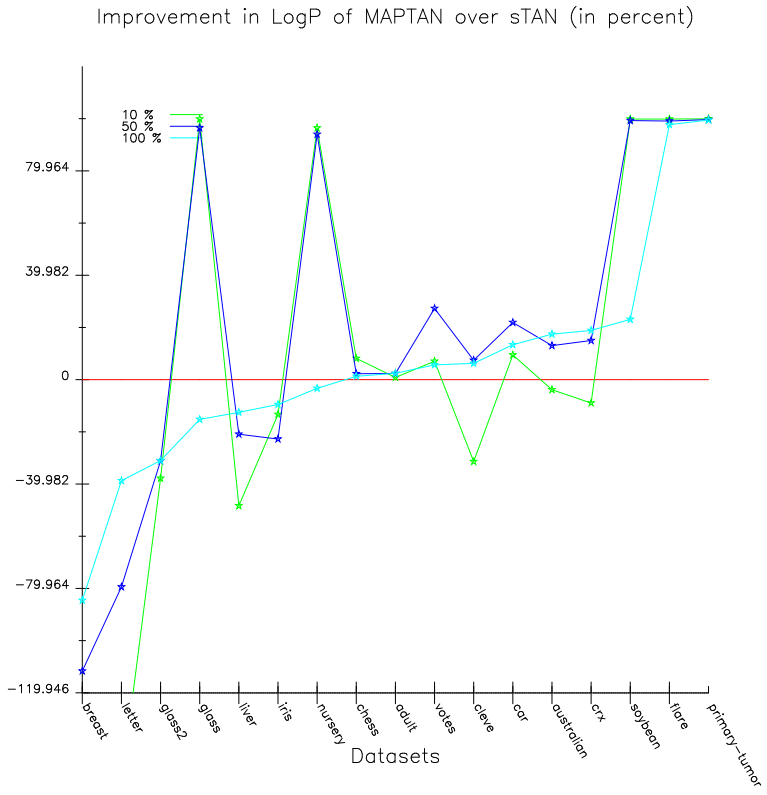


Figure 9.2: Comparison of MAPTAN and STAN *LogScore*

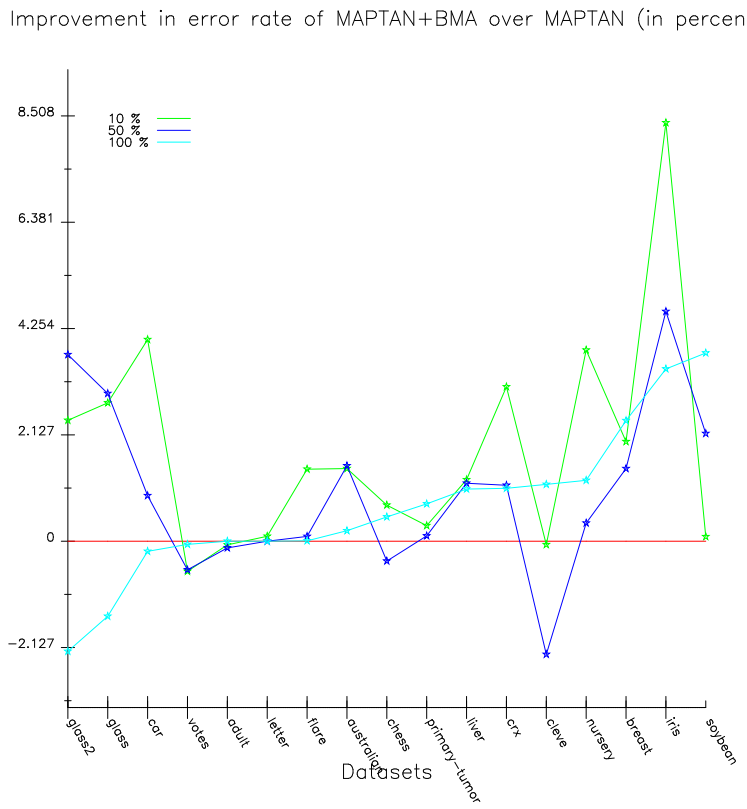


Figure 9.3: Comparison of MAPTAN+BMA and MAPTAN error rate

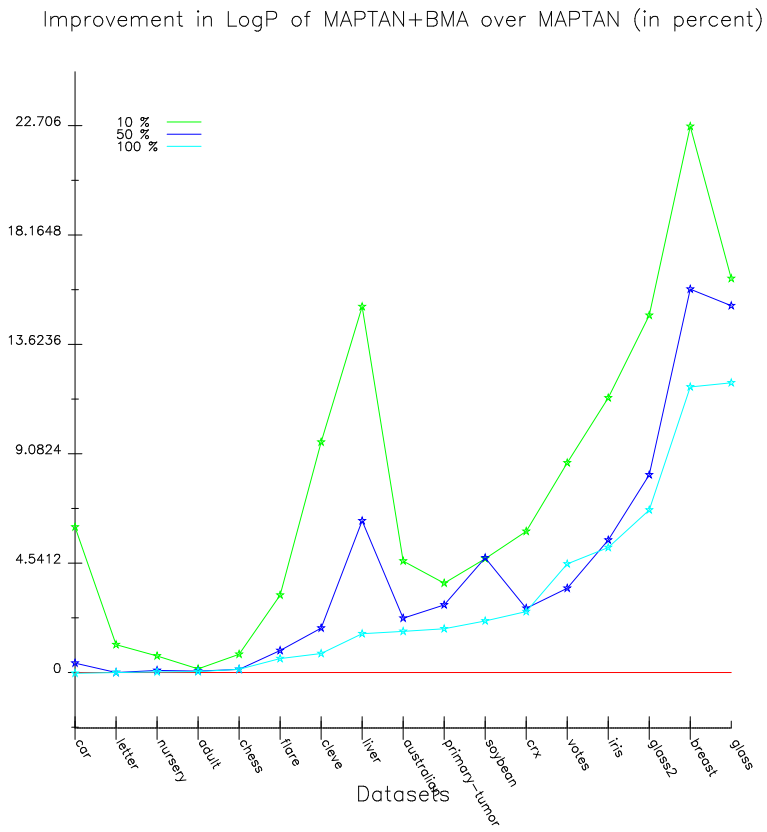


Figure 9.4: Comparison of MAPTAN+BMA and MAPTAN *LogScore*

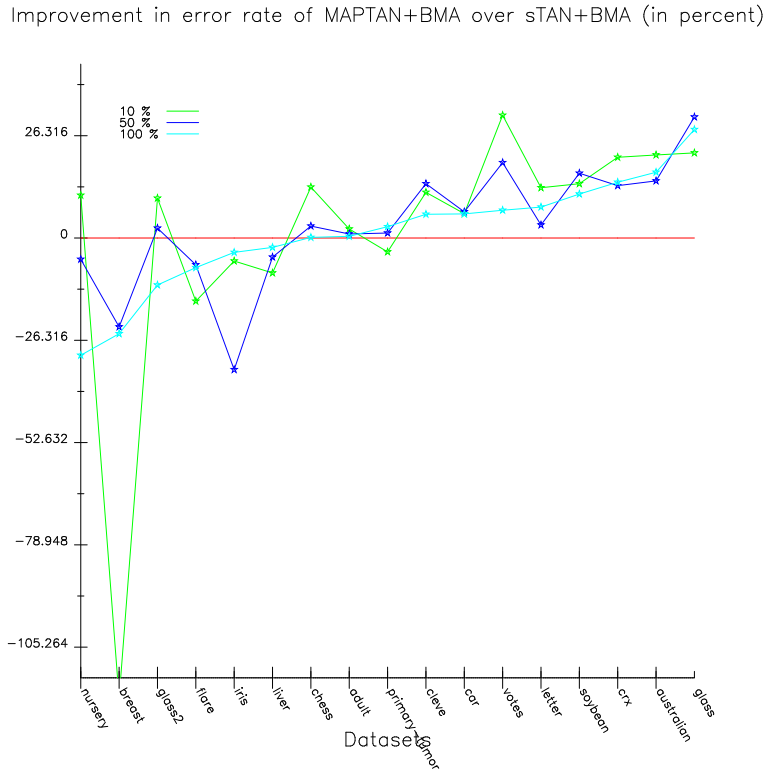


Figure 9.5: Comparison of MAPTAN+BMA and STAN+BMA error rate

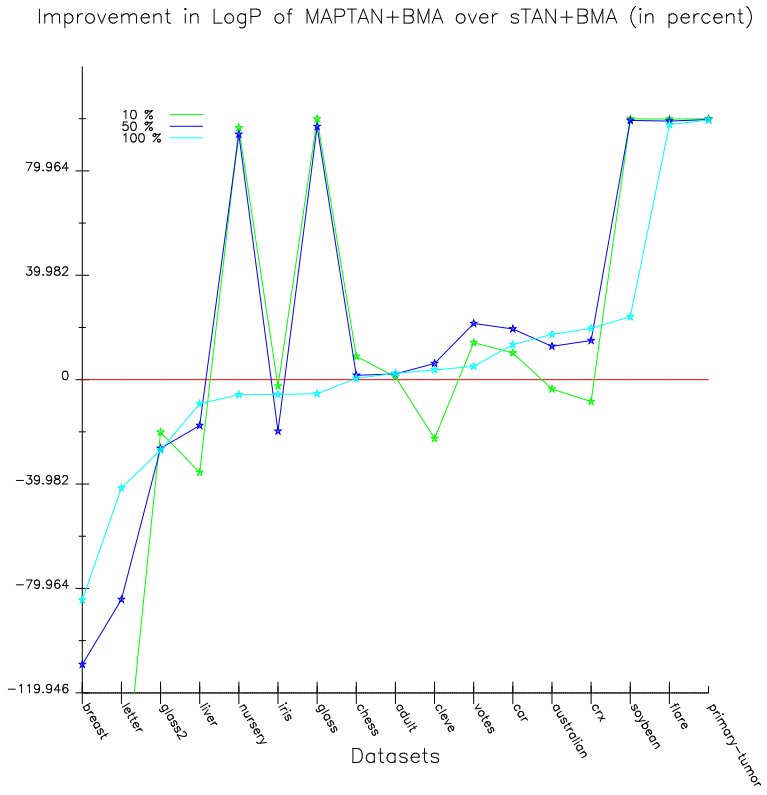


Figure 9.6: Comparison of MAPTAN+BMA and STAN+BMA *LogScore*

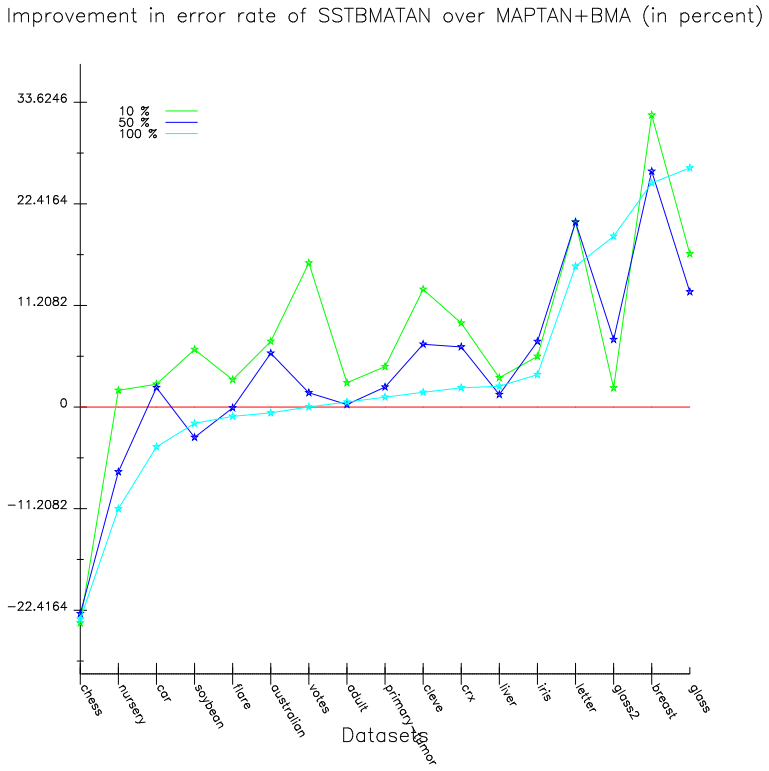


Figure 9.7: Comparison of SSTBMATAN and MAPTAN+BMA error rate

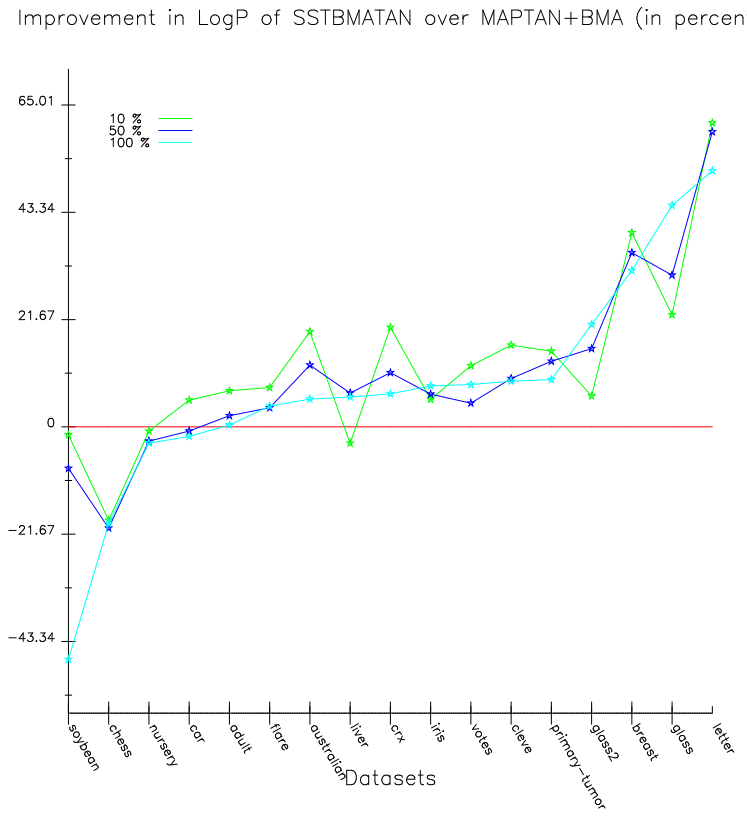


Figure 9.8: Comparison of SSTBMATAN and MAPTAN+BMA *LogScore*

MAPTAN+BMA **vs** MAPTAN

MAPTAN+BMA improves MAPTAN *LogScore* in a statistically significant way for most datasets. When little data is available, this improvement translates in an improvement in error rate. This can be appreciated in figures 9.3 and 9.4. After performing a 5% statistical significance t-test, we have that MAPTAN+BMA error rate is significantly better than MAPTAN's for 6, 0 and 2 datasets with 10%, 50% and 100% of the learning data respectively, whilst MAPTAN error rate is never better than MAPTAN+BMA's in a statistically significant way. *LogScore* results favor MAPTAN+BMA more clearly. MAPTAN+BMA *LogScore* is significantly better than MAPTAN for 16, 13 and 12 datasets with 10%, 50% and 100% of the learning data respectively, whilst MAPTAN *LogScore* never improves SSTBMATAN's in a statistically significant way.

MAPTAN+BMA **vs** STAN+BMA

MAPTAN+BMA improves STAN+BMA error rate and *LogScore* in a statistically significant way for many datasets. This can be appreciated in figures 9.5 and 9.6. After performing a 5% statistical significance t-test, we have that MAPTAN+BMA error rate is significantly better than STAN+BMA for 10, 10 and 8 datasets with 10%, 50% and 100% of the learning data respectively, whilst STAN+BMA error rate is only better than MAPTAN+BMA in a statistically significant way for 5, 4 and 4 datasets . *LogScore* results favor MAPTAN+BMA slightly. MAPTAN+BMA *LogScore* is significantly better than STAN+BMA for 7, 9 and 7 datasets with 10%, 50% and 100% of the learning data respectively, whilst STAN+BMA *LogScore* is only better MAPTAN+BMA in a statistically significant way for 5 datasets independently of the amount of data.

SSTBMATAN **vs** MAPTAN+BMA

SSTBMATAN improves MAPTAN+BMA error rate and *LogScore* in a statistically significant way. This improvement is clearer when little data is available. This can be appreciated in figures 9.7 and 9.8. After performing a 5% statistical significance t-test, we have that SSTBMATAN error rate is significantly better than MAPTAN+BMA for 14, 9 and 4 datasets with 10%, 50% and 100% of the learning data respectively, whilst MAPTAN+BMA error rate is better than SSTBMATAN in a statistically significant way only for 1, 2 and 3 datasets. *LogScore* results favor SSTBMATAN more clearly. SSTBMATAN *LogScore* is significantly better than MAPTAN+BMA for 13, 12 and 12 datasets with 10%, 50% and 100% of the learning data respectively, whilst MAPTAN+BMA *LogScore* is better than SSTBMATAN in a statistically significant way for 2, 3 and 4 datasets respectively.

9.4 Conclusions and future work

In this chapter we have seen that under a decomposable distribution over trees it is possible to efficiently determine the MAP tree and the set of k MAP trees and

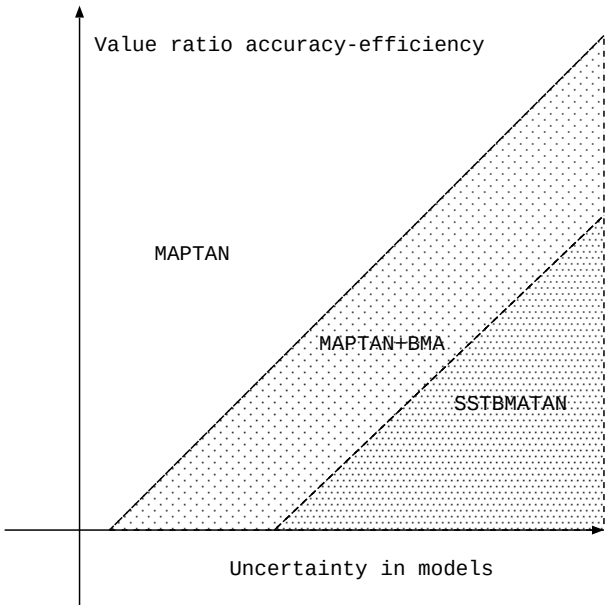


Figure 9.9: Selecting between SSTBMATAN, MAPTAN+BMA and MAPTAN

their relative probability weights. We have extended these results for TAN's and have used them to construct two new classifiers: MAPTAN and MAPTAN+BMA. We have provided empirical results showing that both classifiers improve over their corresponding equivalents seen in chapter 7. Our empirical results do also show that SSTBMATAN is still the most accurate classifier based on the TAN model. From a practical point of view, selecting when to use SSTBMATAN, MAPTAN+BMA or MAPTAN depends mainly on two factors: the amount of uncertainty a posteriori in the models we expect to have and the ratio between the value of accuracy and the value of efficiency for the user. We can see a qualitative sketch of when to choose each classifier in figure 9.9. For any value of the ratio, we will choose MAPTAN when uncertainty a posteriori in models is low, SSTBMATAN when it is high and MAPTAN+BMA inbetween. If learning takes place in an environment where accuracy is much more important than efficiency, then our threshold in uncertainty to use MAPTAN+BMA and SSTBMATAN will be lower. If learning takes place in an environment where efficiency is much more important than accuracy, then MAPTAN will be our choice most of the times unless uncertainty in models is very high.

9.4.1 Future work

Since the amount of uncertainty a posteriori in the models can be measured after the learning step has been done, and the computational overload for finding the

k MAP TAN models is low, it is possible to construct a classifier that is able to make the selection automatically. This classifier could receive a threshold in the amount of uncertainty in model selection. From the k MAP weights the classifier can easily calculate a lower bound on the amount of uncertainty in model selection using the k MAP models, and the single MAP model and select the classifier to be used based in this bound. Furthermore, this evaluation can be performed periodically (each 1000 instances for example). Assuming the dataset is i.i.d., once the uncertainty in a single MAP model is under the threshold, the learning algorithm can assume that the structure has been learnt and from there on its learning time will be $\mathcal{O}(n)$, instead of on the current $\mathcal{O}(n^2)$. This means that we can use the results in this chapter to construct an almost linear TAN learning algorithm.

In chapter 8 we showed that SSTBMATAN performed better than TBMATAN empirically and argued that this was possibly due to the fact that TBMATAN assumes that the distribution generating the data is coming from a TAN, so possibly it overweights the probabilities assigned to the most probable models. A possible test for this hypothesis will be to use the modified weights of SSTBMATAN in MAPTAN+BMA and comparing the performance of the structure stubborn MAPTAN+BMA against that of MAPTAN+BMA. Developing these ideas and tests remains as future work.

Chapter 10

Conclusions

The world is round and the place which may seem like the end may also be only the beginning

Ivy Baker

Along the preceding chapters we have presented several improvements to Bayesian network classifiers. In the first section of this chapter we review the contributions in the thesis. After that, the present the list of international refereed publications produced by the work presented in the thesis. In the last section we try to foresee future applications of these contributions and future research lines.

10.1 Main contributions and its relevance

In this thesis we have proposed different ways to improve Bayesian network classifiers. In the following sections we review each contribution shortly, summarize its relevance and point out possible future applications.

10.1.1 A parallelizable discretization method

Many classifiers can only be applied in discrete domains. Discretization algorithms allow for the application of such classifiers in continuous domains. We have introduced a discretization algorithm based on a distance between partitions that is equivalent in terms of accuracy with state of the art discretization methods and that can be parallelized efficiently. This discretization algorithm could be beneficial for large classification tasks whenever a parallel machine is available.

10.1.2 Qualitative influences and synergies

Bayesian reasoning has been criticized as hard to explain and understand. We have introduced two concepts which help increasing this understandability: influences and synergies. We have also seen that understandability can be further improved by making these concepts qualitative. Qualitative influences and synergies can be used to give acceptable explanations for Bayesian reasoning results. They can also be used in the construction of new, more easily interpretable classifiers.

10.1.3 First Order Qualitative Bayesian Classifier

We have introduced a very simple and easily interpretable classifier based on qualitative influences.

This classifier can be used to quickly explore an unknown dataset to grasp the most relevant interactions with the class, before a more sophisticated and accurate classifier is applied.

10.1.4 Second Order Qualitative Bayesian Classifier

We have introduced an slightly more complex but still easily interpretable classifier based on both qualitative influences and synergies. This classifier has shown to be competitive in accuracy with classifiers such as Naive Bayes while having more easily interpretable results.

This classifier can be used to get information about the most relevant synergies between pairs of attributes and the class, providing a more detailed information about the dataset than the First Order Qualitative Naive Bayesian Classifier.

10.1.5 Naive distributions

We have introduced Naive distributions, which are a family of probability distributions over the set of naive Bayes models. They allow to deal with uncertainty in model selection when we now that our data follows a naive Bayes model. In our opinion, they constitute the most reasonable way to provide information about a domain prior to the application of an algorithm for learning naive Bayes models. They also provide the foundation for the indifferent naive Bayes classifier.

10.1.6 The indifferent naive Bayes classifier

We have introduced the indifferent naive Bayes classifier. The indifferent naive Bayes classifier is the classifier resulting from the assumption that the data follows a naive Bayes model and that the prior over models follows a naive distribution. The development presented here for the indifferent naive Bayes classifier, constitutes an example of a more widely applicable procedure for the construction of classifiers. In fact, the same procedure is followed in the development of

the Tractable Bayesian Model Averaging of Tree Augmented Naive Bayes. The indifferent naive Bayes classifier is more accurate than other classifiers based on naive Bayes models and this improvement is specially significant for datasets with a small number of instances.

The indifferent naive Bayes can be applied wherever Naive Bayes has been applied because its application is likely to result in a small increase in accuracy.

10.1.7 Empirical local Bayesian model averaging of TAN

We have introduced STAN+BMA, a classification algorithm based on applying empirical local Bayesian model averaging to TAN models. The algorithm has been shown to improve TAN accuracy.

This classifier is improved both in complexity and classification accuracy by MAPTAN+BMA (see section 10.1.11), , so we foresee no future application for this result. On the contrary, empirical Bayesian model averaging can be applied to other probabilistic classifiers as a wrapper to deal with model uncertainty.

10.1.8 Decomposable distributions over TAN models

We have introduced decomposable distributions over TAN models, which are a family of probability distributions over the space of TAN models, and an extension for TAN models of decomposable distributions as introduced by Meila and Jaakola (Meila and Jaakkola, 2000b). Decomposable distributions over TANs are conjugate to TAN models. They provide a reasonable way to encode information about a domain prior to the application of an algorithm for learning a TAN model from data. They also provide the foundation for the TBMATAN, MAPTAN and MAPTAN+BMA classifiers.

10.1.9 Tractable Bayesian model averaging of TAN

We have introduced TBMATAN and SSTBMATAN, classification algorithms based on the TAN model, decomposable distributions over TAN models, Bayesian model averaging, and the principle of indifference. These algorithms are based on the fact that decomposable distributions over TANs allow the computation of the averaging over TAN structures in polynomial time. The SSTBMATAN algorithm improves STAN accuracy and is, to the best of our knowledge, the best algorithm reported with a learning time linear on the number of observations in the dataset.

The classifier is specially best suited for datasets with a very large number of observations and a relatively small number of attributes and in tasks where high accuracy is needed and the amount of unclassified data is not overwhelming

10.1.10 Maximum a posteriori TAN classifier

We have introduced MAPTAN, a classification algorithm based on the TAN model, decomposable distributions over TAN models and the principle of indifference.

This algorithm is based on the fact that decomposable distributions over TANs allow the computation of maximum a posteriori TAN structure and parameters in polynomial time. This algorithm improves STAN accuracy, provides theoretical support for the use of softening in TAN induction and has equivalent time complexity.

This algorithm can be applied wherever STAN has been applied because its application is likely to result in an increase in accuracy at no complexity cost.

10.1.11 Maximum a posteriori local Bayesian model averaging of TAN

We have introduced MAPTAN+BMA, a classification algorithm based on the TAN model, decomposable distributions over TAN models, local Bayesian model averaging, and the principle of indifference. This algorithm is based on the fact that decomposable distributions over TANs allow the computation of the k maximum a posteriori TAN structures and parameters in polynomial time. This algorithm improves the accuracy of the probabilities assigned to the class by MAPTAN at a reasonable time complexity cost.

This algorithm can be applied wherever STAN has been applied and we are interested in increasing accuracy even if this means a small increase in time complexity. The TAN classifiers based on decomposable distributions MAPTAN, MAPTAN+BMA and SSTBMATAN can be seen as three different alternatives for different tradeoffs between accuracy and complexity. If time constraints are very important, then we should use MAPTAN. If accuracy is the most important characteristic, then we should use SSTBMATAN. In the space betweeen these two alternatives we can use MAPTAN+BMA.

10.2 Publication list

The work presented in this thesis has also produced the following refereed publications:

- Cerquides, J. and López de Màntaras, R. Proposal and empirical comparison of a parallelizable distance-based discretization method. In Heckerman, D., Mannila, H., Pregibon, D., and Uthurusamy, R., editors, Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, pages 139-142.

- López de Màntaras, R., Cerquides, J., and Garcia, P. (1998). Comparing Information-Theoretic Attribute Selection Measures: A Statistical Approach. AI Communications, 11(2), pages 91-100.

- Cerquides, J. and López de Màntaras, R. Fuzzy metaqueries for guiding the Discovery Process in KDD. In Proceedings of the Sixth International Conference on Fuzzy Systems, volume Volume III, pages 1555-1559.

- J. Cerquides, R. López de Màntaras. On the Usefulness of Component-Based Architectures for Knowledge Discovery. In the Proceedings of the "Primer Congrés Català d'Intel.ligència Artificial (CCIA'98)", Tarragona, Spain, October 21-23, pages 177-180.

- Cerquides, J. and López de Màntaras, R. A first analysis of qualitative influences and synergies. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, AAAI-98 Student Abstracts Session.

- Cerquides, J. and López de Màntaras, R. Knowledge discovery with qualitative influences and synergies. In Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD-98), volume 1510 of LNAI, pages 273-281. Springer.

- Cerquides, J. and López de Màntaras, R. A New Approach to Rule Interest Measures. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, AAAI-98 Student Abstracts Session.

- Cerquides, J. Applying General Bayesian Techniques to Improve TAN Induction. In Proceedings of the International Conference on Knowledge Discovery and Data Mining, KDD99, pages 292-296.

- Cerquides, J. and López de Màntaras, R. The indifferent naive bayes classifier. In Proceedings of the 16th International FLAIRS Conference, FLAIRS 2003, pages 341-345.

- Cerquides, J. and López de Màntaras, R. Tractable Bayesian Learning of Tree Augmented Naive Bayes Models. In Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), pages 75-82.

10.3 Where to go from here?

The contributions presented in section 10.1 support the thesis that the joint application of Bayesian model averaging and a careful selection of the prior probability distribution over the set of models, following objective Bayesian techniques whenever it is possible, can provide significant improvements to classification algorithms. We are convinced that the application of these techniques to other probabilistic models could give equally beneficial results. This is specially clear for the case of classifiers based on models with an underlying tree-like dependency structure, where we are convinced that an extension of decomposable distributions can be easily found.

Some of the contributions in the thesis are new classifiers which are to be included in the "palette" of classifiers to be used by a user of automatic classification. In this sense, it would be interesting to clarify how to increase the interpretability of the results of the classifiers resulting from applying Bayesian model averaging. Better understanding and explaining Bayesian model averaging results appears as an important area of work if we would like to be able

to take full advantage in real life applications of the improvements in accuracy provided by the application of Bayesian model averaging.

Maybe it is also possible to find versions of the classifiers reported here with a much more reduced complexity by applying concentration bounds. Concentration bounds, such as Chernoff bounds, give a bound on the quality of a probability estimate given the amount of data that was used in the estimation. Given concrete quality constraints, we could invert the process and calculate the amount of data needed to get to the desired quality constraints. Further developing this idea and how to apply it for classifiers based on Bayesian model averaging will benefit in lowering the complexity of the classifiers presented in this thesis.

Appendix A

Mathematical developments for the Indifferent Bayesian Classifier

A.1 Preliminaries

A.1.1 A multiple variable constrained integral

Let δ be a function defined as:

$$\delta(x) = \begin{cases} 1 & x = 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.1}$$

Given a real vector $\mathbf{m} = (m_1, \ldots, m_V)$ and a real number r the value of the definite multiple integral

$$\int_{\gamma_\cdot} \cdots \int \prod_{i=1}^{V} \gamma_i^{m_i} \delta\Big(\sum_{i=1}^{V} \gamma_i - r\Big) d\gamma_\cdot \tag{A.2}$$

where the limits of integration range from 0 to ∞ for every γ_i is:

$$\int_{\gamma_\cdot} \cdots \int \prod_{i=1}^{V} \gamma_i^{m_i} \delta\Big(\sum_{i=1}^{V} \gamma_i - r\Big) d\gamma_\cdot = \frac{\prod_{i=1}^{V} \Gamma(m_i + 1)}{\Gamma(M + V)} r^{(M+V-1)} \tag{A.3}$$

where $M = \sum_{i=1}^{V} m_i$.

Proof: The integral can be solved by means of Laplace transforms as can be seen in (Jaynes, 1996) page 1814.

□

A.1.2 A bit of notation

Given a classified discrete domain $\Omega_C = \{A_1, \ldots, A_n, C\}$, let $S = (s_1, \ldots, s_n)$ be an unclassified observation over Ω_C, C the class attribute, s_C and c values of the class attribute, i an attribute index for denoting A_i and v a value of the attribute A_i. We can define:

$$1_C^{S,s_C}(c) = \begin{cases} 1 & c = s_C \\ 0 & \text{otherwise} \end{cases} \tag{A.4}$$

$$1_{i,C}^{S,s_C}(k,c) = \begin{cases} 1 & k = s_i \wedge c = s_C \\ 0 & \text{otherwise} \end{cases} \tag{A.5}$$

It is easy to see that:

$$\sum_{v \in A_i} 1_{i,C}^{S,s_C}(v,c) = 1_C^{S,s_C}(c) \tag{A.6}$$

$$\sum_{c \in C} 1_C^{S,s_C}(c) = 1 \tag{A.7}$$

A.2 Calculating probabilities with naive distributions

Assume that our data is generated by a naive Bayes model and that $P(M|\xi)$, the prior probability distribution over the set of models, follows a naive distribution with hyperparameter set $\mathbf{N}'$. We can calculate the probability of an observation S, s_C given ξ by averaging over the set of naive Bayes models $\mathcal{M}$:

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\xi) = \int_{M \in \mathcal{M}} P(\mathcal{V} = S, \mathcal{C} = s_C|M) P(M|\xi) \tag{A.8}$$

From the definition of naive distribution in equation 6.10 we get $P(M|\xi)$. From the definition of naive Bayes model in equation 6.3 we get $P(\mathcal{V} = S, \mathcal{C} = s_C|M)$. Substituting these two equations into equation A.8 and making use of the notation introduced in section A.1.2 we get:

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\xi) =$$

$$\mathcal{K} \int_{M \in \mathcal{M}} \left(\prod_{c \in C} \alpha_c^{N'_C(c) + 1_C^{S,s_C}(c)} \prod_{i=1}^{n} \prod_{v \in A_i} \left(\frac{\phi_{i,v,c}}{\alpha_c} \right)^{N'_{i,C}(v,s_C) + 1_{i,C}^{S,s_C}(v,c)} \right) dM =$$

$$\mathcal{K} \int_{M \in \mathcal{M}} \left(\prod_{c \in C} \alpha_c^{(1-n)(N'_C(c) + 1_C^{S,s_C}(c))} \prod_{i=1}^{n} \prod_{v \in A_i} {\phi_{i,v,c}}^{N'_{i,C}(v,c) + 1_{i,C}^{S,s_C}(v,c)} \right) dM \tag{A.9}$$

Since our model imposes that $\sum_{c \in C} \alpha_c = 1$ and $\forall i \ \forall c \quad \alpha_c = \sum_{v \in A_i} \phi_{i,v,c}$ we can decompose the previous integral in a set of integrals where our model parameters

range from 0 to ∞ encoding our model restrictions as δ functions. From now on the integral signs should be understood as ranging from 0 to ∞. Hence, defining

$$\mathcal{B}(i,c) = \int_{\phi_{i,.,c}} \cdots \int \prod_{v \in A_i} {\phi_{i,v,c}}^{N'_{i,C}(v,c)+1_{i,C}^{S,s_C}(v,c)} \; \delta\big(\sum_{v \in A_i} \phi_{i,v,c} - \alpha_c\big) d\phi_{i,.,c} \quad \text{(A.10)}$$

we can express A.9 as:

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\xi) =$$
$$\mathcal{K} \int_{\alpha_.} \cdots \int \left(\prod_{c \in C} \alpha_c^{(1-n)(N'_C(c)+1_C^{S,s_C}(c))} \prod_{i=1}^{n} \mathcal{B}(i,c) \; \delta\Big(\sum_{c \in C} \alpha_c - 1\Big) \right) d\alpha_. \quad \text{(A.11)}$$

Applying the result in equation A.3 to $\mathcal{B}(i,c)$ we have

$$\mathcal{B}(i,c) = \frac{\prod_{v \in A_i} \Gamma\left(N'_{i,C}(v,c) + 1_{i,C}^{S,s_C}(v,c) + 1\right)}{\Gamma\left(N'_C(c) + 1_C^{S,s_C}(c) + \#A_i\right)} \alpha_c^{N'_C(c)+1_C^{S,s_C}(c)+\#A_i-1} \quad \text{(A.12)}$$

Then, substituting in A.11 and grouping:

$$p(\mathcal{C} = c_l, \mathcal{V} = S|\mathcal{D}, \xi) = \mathcal{K} \prod_{c \in C} \prod_{i=1}^{n} \frac{\prod_{v \in A_i} \Gamma\left(N'_{i,C}(v,c) + 1_{i,C}^{S,s_C}(v,c) + 1\right)}{\Gamma\left(N'_C(c) + 1_C^{S,s_C}(c) + \#A_i\right)} \times$$
$$\times \int_{\alpha_.} \cdots \int \prod_{c \in C} \alpha_c^{N'_C(c)+1_C^{S,s_C}(c)+\sum_{i=1}^{n}(\#A_i)-n} \delta\Big(\sum_{c \in C} \alpha_c - 1\Big) d\alpha_. \quad \text{(A.13)}$$

Applying A.3 again we get

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\xi) = \mathcal{K} \prod_{c \in C} \prod_{i=1}^{n} \frac{\prod_{v \in A_i} \Gamma\left(N'_{i,C}(v,c) + 1_{i,C}^{S,s_C}(v,c) + 1\right)}{\Gamma\left(N'_C(c) + 1_C^{S,s_C}(c) + \#A_i\right)} \times$$
$$\times \frac{\prod_{c \in C} \Gamma\left(N'_C(c) + 1_C^{S,s_C}(c) + \sum_{i=1}^{n}(\#A_i) - n\right)}{\Gamma\left(N' + 2 + \#C \cdot [\sum_{i=1}^{n}(\#A_i) - n + 1]\right)} \quad \text{(A.14)}$$

We can calculate the *Bayes factor* dividing $P(\mathcal{V}=S,\mathcal{C}=s_C|\xi)$ by the probability of a reference class s'_C:

$$\frac{P(\mathcal{V}=S,\mathcal{C}=s_C|\xi)}{P(\mathcal{V}=S,\mathcal{C}=s'_C|\xi)} = \prod_{i=1}^{n}\left[\frac{N'_{i,C}(s_i,s_C)+1}{N'_C(s_C)+\#A_i}\left(\frac{N'_{i,C}(s_i,s'_C)+1}{N'_C(s'_C)+\#A_i}\right)^{-1}\right]\times \\ \times\frac{N'_C(s_C)+1+\sum_{i=1}^{n}(\#A_i)-n}{N'_C(s'_C)+1+\sum_{i=1}^{n}(\#A_i)-n} \quad (A.15)$$

Since equation A.15 is symmetrical with respect to s_C and S'_C, we can infer from it that:

$$P(\mathcal{V}=S,\mathcal{C}=s_C|\xi) = \mathcal{K}'\left(N'_C(s_C)+1+\sum_{i=1}^{n}\#A_i-n\right)\prod_{i=1}^{n}\frac{N'_{i,C}(s_i,s_C)+1}{N'_C(s_C)+\#A_i} \quad (A.16)$$

where $\mathcal{K}'$ is a normalization constant. Given that $P(\mathcal{C}=s_C|\mathcal{V}=S,\xi) = \frac{P(\mathcal{V}=S,\mathcal{C}=s_C|\xi)}{P(\mathcal{V}=S|\xi)}$, we also have that

$$P(\mathcal{C}=s_C|\mathcal{V}=S,\xi) = \mathcal{K}''\left(N'_C(s_C)+1+\sum_{i=1}^{n}\#A_i-n\right)\prod_{i=1}^{n}\frac{N'_{i,C}(s_i,s_C)+1}{N'_C(s_C)+\#A_i} \quad (A.17)$$

where $\mathcal{K}''=\frac{\mathcal{K}'}{P(\mathcal{V}=S|\xi)}$ is another normalization constant. Given that

$$\sum_{s_C\in C}P(s_C|\mathcal{V}=S,\xi)=1 \quad (A.18)$$

we have that

$$\mathcal{K}''=\frac{1}{\sum_{c\in C}\left[\left(N'_C(c)+1+\sum_{i=1}^{n}\#A_i-n\right)\prod_{i=1}^{n}\frac{N'_{i,C}(s_i,c)+1}{N'_C(c)+\#A_i}\right]} \quad (A.19)$$

This completes the result exposed in section 6.2.1.

□

A.3 Learning with naive distributions

Assume that our data is generated by a naive Bayes model, that $P(M|\xi)$ follows a naive distribution with hyperparameter set $\mathbf{N}'$ and that $\mathcal{D}$ is a dataset containing independent identically distibuted complete observations over a classified discrete doman Ω_C. We can calculate the posterior probability over models given $\mathcal{D}$ and ξ by applying Bayes theorem over $p(M|\mathcal{D},\xi)$:

$$p(M|\mathcal{D},\xi)=p(\mathcal{D}|M,\xi)\frac{p(M|\xi)}{p(\mathcal{D}|\xi)} \quad (A.20)$$

A.3.1 Computing the normalization constant

The denominator of equation A.20 is a normalization constant that can be expanded as:

$$p(\mathcal{D}|\xi) = \int_{M\in\mathcal{M}} p(\mathcal{D}|M,\xi)p(M|\xi)dM \tag{A.21}$$

Since the dataset contains indepedent identically distributed complete observations, we can compute $p(\mathcal{D}|M,\xi)$ by a repeated application of equation 6.3:

$$P(\mathcal{D}|M,\xi) = \prod_{c\in C}\alpha_c^{N_C(c)}\prod_{i=1}^{n}\prod_{v\in A_i}\left(\frac{\phi_{i,v,c}}{\alpha_c}\right)^{N_{i,C}(v,c)} \tag{A.22}$$

Substituting equations A.22 and 6.10 into equation A.21 we get:

$$p(\mathcal{D}|\xi) = \mathcal{K}\int_{M\in\mathcal{M}}\prod_{c\in C}\alpha_c^{N_C(c)+N'_C(c)}\prod_{i=1}^{n}\prod_{v\in A_i}\left(\frac{\phi_{i,v,c}}{\alpha_c}\right)^{N_{i,C}(v,c)+N'_{i,C}(v,c)} dM \tag{A.23}$$

As in the previous section, integrating over models can be replaced by integrating over parameters adding δ constraints. Doing this and applying twice the result in equation A.3 similarly to what we did in the previous section we get:

$$\begin{aligned} p(\mathcal{D}|\xi) = \mathcal{K}\prod_{c\in C}\prod_{i=1}^{n}\frac{\prod_{v\in A_i}\Gamma\left(N_{i,C}(v,c)+N'_{i,C}(v,c)+1\right)}{\Gamma\left(N_C(c)+N'_C(c)+\#A_i\right)}\times \\ \times\frac{\prod_{c\in C}\Gamma\left(N'_C(c)+N_C(c)+\sum_{i=1}^{n}(\#A_i)-n+1\right)}{\Gamma\left(N'+N+\#C\cdot[\sum_{i=1}^{n}(\#A_i)-n+1]+1\right)} \end{aligned} \tag{A.24}$$

A.3.2 Computing the posterior distribution

In order to complete the calculation of the posterior distribution we only have to substitute equations A.22, 6.10 and A.24 into equation A.20 getting:

$$\begin{aligned} p(M|\mathcal{D},\xi) = \prod_{c\in C}\prod_{i=1}^{n}\frac{\Gamma\left(N_C(c)+N'_C(c)+\#A_i\right)}{\prod_{v\in A_i}\Gamma\left(N_{i,C}(v,c)+N'_{i,C}(v,c)+1\right)}\times \\ \times\frac{\Gamma\left(N'+N+\#C\cdot[\sum_{i=1}^{n}(\#A_i)-n+1]+1\right)}{\prod_{c\in C}\Gamma\left(N'_C(c)+N_C(c)+\sum_{i=1}^{n}(\#A_i)-n+1\right)}\times \\ \times\prod_{c\in C}\alpha_c^{N_C(c)+N'_C(c)}\prod_{i=1}^{n}\prod_{v\in A_i}\left(\frac{\phi_{i,v,c}}{\alpha_c}\right)^{N_{i,C}(v,c)+N'_{i,C}(v,c)} \end{aligned} \tag{A.25}$$

which is a naive distribution with hyperparameter set $\mathbf{N}'^{*}$ where $N'^{*}_{i,C}(v,c) = N_{i,C}(v,c) + N'_{i,C}(v,c)$.

Appendix B

Mathematical developments for the Tractable Bayesian Model Averaging of Tree Augmented Naive Bayes Classifiers

B.1 Preliminaries

In this appendix we introduce three results that will be needed in the further development and then in appendix B.3 we prove the results in sections 8.2.2 and 8.2.3.

B.1.1 The matrix tree theorem

Let $G = (V, E)$ be a multigraph and denote by $a_{u,v} = a_{v,u}$ the number of undirected edges between vertices u and v. Then the number of all spanning trees of G is given by the value of the determinant obtained from the following matrix by removing row u and column v.

$$A = \begin{bmatrix} \deg v_1 & -a_{1,2} & -a_{1,3} & \dots & a_{1,n} \\ -a_{2,1} & \deg v_2 & -a_{2,3} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots & \dots \\ -a_{n,1} & -a_{n,2} & -a_{n,3} & \dots & \deg v_n \end{bmatrix} \tag{B.1}$$

Proof: See (West, 1999; Rubey, 2000).

□

B.1.2 The matrix tree theorem for decomposable distributions

Let P(E) be a distribution over spanning tree structures defined by equations 8.3 and 8.4. Then the normalization constant Z_β is equal to $|Q(\boldsymbol{\beta})|$ with $Q(\boldsymbol{\beta})$ being the first (n-1) lines and columns of the matrix $\overline{Q}(\boldsymbol{\beta})$ given by:

$$\overline{Q}_{u,v}(\boldsymbol{\beta}) = \overline{Q}_{v,u}(\boldsymbol{\beta}) = \begin{cases} -\beta_{u,v} & 1 \leq u < v \leq n \\ \sum_{v'=1}^{n} \beta_{v',v} & 1 \leq u = v \leq n \end{cases} \tag{B.2}$$

Proof: See (Meila and Jaakkola, 2000a).

□

B.1.3 A useful result about Dirichlet distributions

Let $D(\theta_1, \ldots, \theta_r; n'_1, \ldots, n'_r)$ be a Dirichlet distribution defined as in equation 3.2. We have that:

$$\begin{aligned} &D(\theta_1, \ldots, \theta_r; n'_1, \ldots, n'_r) \prod_{i=1}^{r} {\theta_i}^{n_i} = \\ &\frac{\Gamma(\sum_{i=1}^{r} n'_i)}{\prod_{i=1}^{r} \Gamma(n'_i)} \frac{\prod_{i=1}^{r} \Gamma(n'_i + n_i)}{\Gamma(\sum_{i=1}^{r} n'_i + n_i)} D(\theta_1, \ldots, \theta_r; n'_1 + n_1, \ldots, n'_r + n_r) \end{aligned} \tag{B.3}$$

and since the Dirichlet distribution is normalized you have that

$$\int_{\theta_1,\ldots,\theta_r} \cdots \int D(\theta_1, \ldots, \theta_r; n'_1, \ldots, n'_r) \prod_{i=1}^{r} {\theta_i}^{n_i} = \frac{\Gamma(\sum_{i=1}^{r} n'_i)}{\prod_{i=1}^{r} \Gamma(n'_i)} \frac{\prod_{i=1}^{r} \Gamma(n'_i + n_i)}{\Gamma(\sum_{i=1}^{r} n'_i + n_i)} \tag{B.4}$$

Proof: By expanding the Dirichlet distribution by means of its definition in equation 3.2, grouping again into a Dirichlet and considering that the Dirichlet

distribution is normalized distribution and hence integrates to one, we have that:

$$\int_{\theta_1,\ldots,\theta_r} \cdots \int D(\theta_1,\ldots,\theta_r;n'_1,\ldots,n'_r) \prod_{i=1}^{r} {\theta_i}^{n_i} \tag{B.5}$$

$$= \int_{\theta_1,\ldots,\theta_r} \cdots \int \frac{\Gamma(\sum_{i=1}^r n'_i)}{\prod_{i=1}^r \Gamma(n'_i)} \prod_{i=1}^{r} {\theta_i}^{n'_i+n_i-1} \tag{B.6}$$

$$= \int_{\theta_1,\ldots,\theta_r} \cdots \int \frac{\Gamma(\sum_{i=1}^r n'_i)}{\prod_{i=1}^r \Gamma(n'_i)} \frac{\prod_{i=1}^r \Gamma(n'_i+n_i)}{\Gamma(\sum_{i=1}^r n'_i+n_i)} D(\theta_1,\ldots,\theta_r;n'_1+n_1,\ldots,n'_r+n_r) \tag{B.7}$$

$$= \frac{\Gamma(\sum_{i=1}^r n'_i)}{\prod_{i=1}^r \Gamma(n'_i)} \frac{\prod_{i=1}^r \Gamma(n'_i+n_i)}{\Gamma(\sum_{i=1}^r n'_i+n_i)} \int_{\theta_1,\ldots,\theta_r} \cdots \int D(\theta_1,\ldots,\theta_r;n'_1+n_1,\ldots,n'_r+n_r) \tag{B.8}$$

$$= \frac{\Gamma(\sum_{i=1}^r n'_i)}{\prod_{i=1}^r \Gamma(n'_i)} \frac{\prod_{i=1}^r \Gamma(n'_i+n_i)}{\Gamma(\sum_{i=1}^r n'_i+n_i)} \tag{B.9}$$

□

B.2 Detailed development for decomposable distributions over trees results

In this appendix we provide the proofs for the results in section 8.1.2.

B.2.1 Calculating probabilities under decomposable distributions over trees

Knowing that $P(M|\xi)$ follows a decomposable distribution over trees with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$ we need to calculate

$$P(\mathcal{X} = x|\xi) = \int_{M\in\mathcal{M}} P(\mathcal{X} = x|M,\xi)P(M|\xi) \tag{B.10}$$

We can calculate the integral over the set of models by calculating the probability of each structure and then performing an addition over the set of structures. In fact, since we have assumed likelihood equivalence and our distribution over directed structures is uniform given the undirected structure, we can work over the set of undirected structures, that is

$$P(\mathcal{X} = x|\xi) = \sum_{E\in\mathcal{E}} P(\mathcal{X} = x|E,\xi)P(E|\xi) \tag{B.11}$$

where $P(E|\xi)$ comes given by:

$$P(E|\xi) = \frac{1}{Z_\beta} \prod_{u,v \in E} \beta_{u,v} \tag{B.12}$$

In order to calculate $P(\mathcal{X} = x|\xi)$ we have to calculate $P(\mathcal{X} = x|E, \xi)$ and then calculate the summmation in equation B.11.

Calculating $P(\mathcal{X} = x|E, \xi)$

Using again likelihood equivalence we can express $P(\mathcal{X} = x|E, \xi)$ as the integral over any directed structure $\overline{E}$ which undirected structure coincides with E:

$$P(\mathcal{X} = x|E, \xi) = \int_{\Theta_{\overline{E}}} \cdots \int P(\mathcal{X} = x|\overline{E}, \Theta_{\overline{E}}) P(\Theta_{\overline{E}}|\overline{E}, \xi) d\Theta_{\overline{E}} \tag{B.13}$$

$P(\mathcal{X} = x|\overline{E}, \Theta_{\overline{E}})$ is determined by the expansion of equation 3.1 taking into account the tree structure.

$$P(\mathcal{X} = x|\overline{E}, \Theta_{\overline{E}}) = \theta_{\rho_{\overline{E}}}(x_{\rho_{\overline{E}}}) \prod_{u,v \in \overline{E}} \theta_{v|u}(x_v, x_u) \tag{B.14}$$

$P(\Theta_{\overline{E}}|\overline{E}, \xi)$ can be expanded from equations 8.5, 8.8 and 8.9 into

$$P(\Theta_{\overline{E}}|\overline{E}, \xi) = D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.)) \prod_{u,v \in \overline{E}} \prod_{i \in A_u} D(\theta_{v|u}(., i); N'_{v,u}(., i)) \tag{B.15}$$

Now we need to calculate the integral in equation B.13 We define:

$$1_i^x(k) = \begin{cases} 1 & k = x_i \\ 0 & \text{otherwise} \end{cases} \tag{B.16}$$

$$1_{i,j}^x(k, l) = \begin{cases} 1 & k = x_i \wedge l = x_j \\ 0 & \text{otherwise} \end{cases} \tag{B.17}$$

It is easy to see that:

$$\sum_{j \in A_v} 1_{v,u}^x(j, i) = 1_u^x(i) \tag{B.18}$$

$$\sum_{i \in A_u} 1_u^x(i) = 1 \tag{B.19}$$

We can use this notation to expand the product $P(\mathcal{X} = x|\overline{E}, \Theta_{\overline{E}}) P(\Theta_{\overline{E}}|\overline{E}, \xi)$ by substituting equations B.14 and B.15 giving:

$$\begin{aligned} P(\mathcal{X} = x|\overline{E}, \Theta_{\overline{E}}) P(\Theta_{\overline{E}}|\overline{E}, \xi) &= D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.)) \prod_{i \in A_{\rho_{\overline{E}}}} \theta_{\rho_{\overline{E}}}(i)^{1_{\rho_{\overline{E}}}^x(i)} \\ &\times \prod_{u,v \in \overline{E}} \prod_{i \in A_u} \left[D(\theta_{v|u}(., i); N'_{v,u}(., i)) \prod_{j \in A_v} \theta_{v|u}(j, i)^{1_{v,u}^x(j,i)} \right] \end{aligned} \tag{B.20}$$

By analyzing equation B.20 we can see that the integral in equation B.13 can be calculated by applying the result in equation B.4 twice. This gives:

$$P(\mathcal{X}=x|E,\xi) = \frac{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}} N'_{\rho_{\overline{E}}}(i))}{\prod_{i\in A_{\rho_{\overline{E}}}} \Gamma(N'_{\rho_{\overline{E}}}(i))} \frac{\prod_{i\in A_{\rho_{\overline{E}}}} \Gamma(N'_{\rho_{\overline{E}}}(i)+1^x_{\rho_{\overline{E}}}(i))}{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}} N'_{\rho_{\overline{E}}}(i)+1^x_{\rho_{\overline{E}}}(i))} \times \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \left[\frac{\Gamma(\sum_{j\in A_v} N'_{v,u}(j,i))}{\prod_{j\in A_v}\Gamma(N'_{v,u}(j,i))} \frac{\prod_{i\in A_v}\Gamma(N'_{v,u}(j,i)+1^x_{v,u}(j,i))}{\Gamma(\sum_{i\in A_v} N'_{v,u}(j,i)+1^x_{v,u}(j,i))} \right] \tag{B.21}$$

This expression can be simplified by applying equations 8.6,8.7,B.18 and B.19 and reorganizing:

$$P(\mathcal{X}=x|E,\xi) = \frac{\Gamma(N')}{\Gamma(N'+1)} \prod_{i\in A_{\rho_{\overline{E}}}} \frac{\Gamma(N'_{\rho_{\overline{E}}}(i)+1^x_{\rho_{\overline{E}}}(i))}{\Gamma(N'_{\rho_{\overline{E}}}(i))} \times \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \left[\frac{\Gamma(N'_u(i))}{\Gamma(N'_u(i)+1^x_u(i))} \prod_{i\in A_v} \frac{\Gamma(N'_{v,u}(j,i)+1^x_{v,u}(j,i))}{\Gamma(N'_{v,u}(j,i))} \right] \tag{B.22}$$

Since the quotient $\frac{\Gamma(N'_*(*)+1^x_*(*))}{\Gamma(N'_*(*))}$ is $N'_*(*)$ if the condition expressed by the $1^x_*(*)$ is satisfied and 1 otherwise we have that:

$$P(\mathcal{X}=x|E,\xi) = \frac{1}{N'} N'_{\rho_{\overline{E}}}(x_{\rho_{\overline{E}}}) \prod_{u,v\in\overline{E}} \left[\frac{N'_{v,u}(x_v,x_u)}{N'_u(x_u)} \right] \tag{B.23}$$

Defining h^x_0 and $h^x_{u,v}$ as in equations 8.19 and 8.20 it is easy to see that multiplying and dividing in equation B.23 by the factor:

$$\prod_{v\in\Omega-\{\rho_{\overline{E}}\}} N'_v(x_v) \tag{B.24}$$

and rearranging we get:

$$P(\mathcal{X}=x|E,\xi) = h^x_0 \; Z_\beta \prod_{u,v\in E} h^x_{u,v} \tag{B.25}$$

and the expression depends only of the undirected structure of the tree.

Adding over Tree Structures

Combining equations B.12, B.25 we get

$$P(\mathcal{X}=x|E,\xi)P(E|\xi) = h^x_0 \prod_{u,v\in E} \beta_{u,v} h^x_{u,v} \tag{B.26}$$

Calculating the summation over structures using the matrix tree theorem for decomposable distributions gives the desired result.

$$P(\mathcal{X} = x|\xi) = h_0^x |Q(\boldsymbol{\beta}\ \mathbf{h}^x)| \tag{B.27}$$

□

B.2.2 Learning under decomposable distributions over trees

Given that $P(M|\xi)$ follows a decomposable distribution over trees with hyper-parameters $\boldsymbol{\beta}, \mathbf{N}'$ we want to calculate $P(M|\mathcal{D}, \xi)$ where $\mathcal{D}$ is an i.i.d. dataset sampled from a tree distribution. Using Bayes rule we get:

$$P(M|\mathcal{D},\xi) = P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D},\xi) = \frac{P(\overline{E}, \Theta_{\overline{E}}|\xi) P(\mathcal{D}|\overline{E}, \Theta_{\overline{E}}, \xi)}{Z_{\mathcal{D}}} \tag{B.28}$$

The prior $P(\overline{E}, \Theta_{\overline{E}}|\xi)$ is calculated combining equations 8.1,B.12, B.15 giving:

$$\begin{aligned} P(\overline{E}, \Theta_{\overline{E}}|\xi) = \frac{1}{Z_\beta} & \prod_{u,v \in \overline{E}} \beta_{u,v} \\ & \times D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.)) \\ & \times \prod_{u,v \in \overline{E}} \prod_{i \in A_u} D(\theta_{v|u}(.,i); N'_{v,u}(.,i)) \end{aligned} \tag{B.29}$$

$P(\mathcal{D}|\overline{E}, \Theta_{\overline{E}}, \xi)$ is the probability that the model generates the data in $\mathcal{D}$. Since $\mathcal{D}$ contains independent identically distributed observations, we have that

$$\begin{aligned} P(\mathcal{D}|\overline{E}, \Theta_{\overline{E}}, \xi) = & \prod_{i \in A_{\rho_{\overline{E}}}} \theta_{\rho_{\overline{E}}}(i)^{N_{\rho_{\overline{E}}}(i)} \\ & \times \prod_{u,v \in \overline{E}} \prod_{i \in A_u} \prod_{j \in A_v} \theta_{v|u}(j,i)^{N_{v,u}(j,i)} \end{aligned} \tag{B.30}$$

Substituting equations B.29 and B.30 into B.28 we get

$$\begin{aligned} P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D},\xi) = & \frac{1}{Z_\beta} \frac{1}{Z_{\mathcal{D}}} \prod_{u,v \in \overline{E}} \beta_{u,v} \\ & \times D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.)) \prod_{i \in A_{\rho_{\overline{E}}}} \theta_{\rho_{\overline{E}}}(i)^{N_{\rho_{\overline{E}}}(i)} \\ & \times \prod_{u,v \in \overline{E}} \prod_{i \in A_u} \left[D(\theta_{v|u}(.,i); N'_{v,u}(.,i)) \prod_{j \in A_v} \theta_{v|u}(j,i)^{N_{v,u}(j,i)} \right] \end{aligned} \tag{B.31}$$

Applying the result in equation B.3 for all the Dirichlets we have that

$$
\begin{aligned}
P(\overline{E},\Theta_{\overline{E}}|\mathcal{D},\xi) = & \frac{1}{Z_\beta}\frac{1}{Z_{\mathcal{D}}}\prod_{u,v\in\overline{E}}\beta_{u,v} \\
& \times \frac{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}} N'_{\rho_{\overline{E}}}(i))}{\prod_{i\in A_{\rho_{\overline{E}}}}\Gamma(N'_{\rho_{\overline{E}}}(i))}\frac{\prod_{i\in A_{\rho_{\overline{E}}}}\Gamma(N'_{\rho_{\overline{E}}}(i)+N_{\rho_{\overline{E}}}(i))}{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}} N'_{\rho_{\overline{E}}}(i)+N_{\rho_{\overline{E}}}(i))} \\
& \times \prod_{u,v\in\overline{E}}\prod_{i\in A_u}\left[\frac{\Gamma(\sum_{j\in A_v} N'_{v,u}(j,i))}{\prod_{j\in A_v}\Gamma(N'_{v,u}(j,i))}\frac{\prod_{j\in A_v}\Gamma(N'_{v,u}(j,i)+N_{v,u}(j,i))}{\Gamma(\sum_{j\in A_v} N'_{v,u}(j,i)+N_{v,u}(j,i))}\right] \\
& \times D(\theta_{\rho_{\overline{E}}}(.);N'_{\rho_{\overline{E}}}(.)+N_{\rho_{\overline{E}}}(.)) \\
& \times \prod_{u,v\in\overline{E}}\prod_{i\in A_u} D(\theta_{v|u}(.,i);N'_{v,u}(.,i)+N_{v,u}(.,i))
\end{aligned}
\tag{B.32}
$$

This expression can be simplified by applying equations 8.6 and 8.7 (and similar ones for N) and reorganizing:

$$
\begin{aligned}
P(\overline{E},\Theta_{\overline{E}}|\mathcal{D},\xi) = & \frac{1}{Z_\beta}\frac{1}{Z_{\mathcal{D}}}\frac{\Gamma(N')}{\Gamma(N'+N)}\prod_{u,v\in\overline{E}}\beta_{u,v} \\
& \times \prod_{i\in A_{\rho_{\overline{E}}}}\frac{\Gamma(N'_{\rho_{\overline{E}}}(i)+N_{\rho_{\overline{E}}}(i))}{\Gamma(N'_{\rho_{\overline{E}}}(i))} \\
& \times \prod_{u,v\in\overline{E}}\prod_{i\in A_u}\left[\frac{\Gamma(N'_u(i))}{\Gamma(N'_u(i)+N_u(i))}\prod_{j\in A_v}\frac{\Gamma(N'_{v,u}(j,i)+N_{v,u}(j,i))}{\Gamma(N'_{v,u}(j,i))}\right] \\
& \times D(\theta_{\rho_{\overline{E}}}(.);N'_{\rho_{\overline{E}}}(.)+N_{\rho_{\overline{E}}}(.)) \\
& \times \prod_{u,v\in\overline{E}}\prod_{i\in A_u} D(\theta_{v|u}(.,i);N'_{v,u}(.,i)+N_{v,u}(.,i))
\end{aligned}
\tag{B.33}
$$

Defining $W_{u,v}$ as appears in equation 8.13, it is easy to see that multiplying and dividing in equation B.33 by the factor:

$$
\prod_{v\in\Omega-\{\rho_{\overline{E}}\}}\prod_{i\in A_v}\frac{\Gamma(N'_v(i)+N_v(i))}{\Gamma(N'_v(i))} \tag{B.34}
$$

and rearranging we get:

$$\begin{aligned} P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D}, \xi) &= \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}}\frac{\Gamma(N')}{\Gamma(N'+N)} \\ &\times \prod_{v\in\Omega}\prod_{i\in A_v}\frac{\Gamma(N'_v(i)+N_v(i))}{\Gamma(N'_v(i))} \\ &\times \prod_{u,v\in\overline{E}} W_{u,v}\beta_{u,v} \\ &\times D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.)+N_{\rho_{\overline{E}}}(.)) \\ &\times \prod_{u,v\in\overline{E}}\prod_{i\in A_u} D(\theta_{v|u}(.,i); N'_{v,u}(.,i)+N_{v,u}(.,i)) \end{aligned} \tag{B.35}$$

It is worth noting that if we use the definition of $W_{u,v}$ given by Meila and Jaakkola (see equation 8.12), our expression will keep a factor that depends on the directed tree structure (concretely on the root) and it would not be possible to continue with our development further on.

In order to have $P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D}, \xi)$ completely determined we need to calculate $Z_\mathcal{D}$. Since we know that

$$\int_{M\in\mathcal{M}} P(M|\mathcal{D}, \xi) = \sum_{E\in\mathcal{E}}\int\cdots\int_{\Theta_{\overline{E}}} P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D}, \xi) = 1 \tag{B.36}$$

We can do this by integrating over the parameters, then summing over the tree structures and finally solving for $Z_\mathcal{D}$. The first step is easy, because Dirichlet distributions are normalized and integrate to 1 giving:

$$\begin{aligned} \int\cdots\int_{\Theta_{\overline{E}}} P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D}, \xi) &= \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}}\frac{\Gamma(N')}{\Gamma(N'+N)} \\ &\times \prod_{v\in\Omega}\prod_{i\in A_v}\frac{\Gamma(N'_v(i)+N_v(i))}{\Gamma(N'_v(i))} \\ &\times \prod_{u,v\in\overline{E}} W_{u,v}\beta_{u,v} \end{aligned} \tag{B.37}$$

The addition over structures can be calculated by means of the matrix tree theorem for decomposable priors, giving

$$\begin{aligned} \sum_{E\in\mathcal{E}}\int\cdots\int_{\Theta_{\overline{E}}} P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D}, \xi) &= \frac{|Q(\boldsymbol{\beta}\mathbf{W})|}{Z_\beta}\frac{1}{Z_\mathcal{D}}\frac{\Gamma(N')}{\Gamma(N'+N)} \\ \times \prod_{v\in\Omega}\prod_{i\in A_v}\frac{\Gamma(N'_v(i)+N_v(i))}{\Gamma(N'_v(i))} &= 1 \end{aligned} \tag{B.38}$$

Solving for $Z_\mathcal{D}$, recalling that $Z_\beta = |Q(\boldsymbol{\beta})|$ we have that

$$Z_\mathcal{D} = \frac{|Q(\boldsymbol{\beta}\mathbf{W})|}{|Q(\boldsymbol{\beta})|}\frac{\Gamma(N')}{\Gamma(N'+N)}\prod_{v\in\Omega}\prod_{i\in A_v}\frac{\Gamma(N'_v(i)+N_v(i))}{\Gamma(N'_v(i))} \tag{B.39}$$

Finally, substituting the result for $Z_{\mathcal{D}}$ in equation B.35 we can see that the posterior is a decomposable distribution over trees with the hyperparameters updated as given by equations 8.10, 8.11 and 8.13:

$$\begin{aligned} P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D}, \xi) = & \frac{1}{|Q(\boldsymbol{\beta}\mathbf{W})|} \prod_{u,v \in \overline{E}} W_{u,v}\beta_{u,v} \\ & \times D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.) + N_{\rho_{\overline{E}}}(.)) \\ & \times \prod_{u,v \in \overline{E}} \prod_{i \in A_u} D(\theta_{v|u}(.,i); N'_{v,u}(.,i) + N_{v,u}(.,i)) \end{aligned} \tag{B.40}$$

□

B.3 Detailed development for decomposable distributions over TANs results

In this appendix we provide the proofs for the results in sections 8.2.2 and 8.2.3.

B.3.1 Calculating probabilities under decomposable distributions over TANs

Knowing that $P(M|\xi)$ follows a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$ we need to calculate

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\xi) = \int_{M \in \mathcal{M}} P(\mathcal{V} = S, \mathcal{C} = s_C|M, \xi)P(M|\xi) \tag{B.41}$$

The development will be parallel to the one in section B.2.1. In this case, we can also work over the set of undirected structures having:

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\xi) = \sum_{E \in \mathcal{E}} P(\mathcal{V} = S, \mathcal{C} = s_C|E, \xi)P(E|\xi) \tag{B.42}$$

where $P(E|\xi)$ comes given by:

$$P(E|\xi) = \frac{1}{Z_\beta} \prod_{u,v \in E} \beta_{u,v} \tag{B.43}$$

Calculating $P(\mathcal{V} = S, \mathcal{C} = s_C|E, \xi)$

Using likelihood equivalence we can express $P(\mathcal{X} = x|E, \xi)$ as the integral over any directed TAN structure $\overline{E}^*$ which undirected structure coincides with E:

$$\begin{aligned} & P(\mathcal{V} = S, \mathcal{C} = s_C|E, \xi) = \\ & \qquad = \int_{\Theta_{\overline{E}^*}} \cdots \int P(\mathcal{V} = S, \mathcal{C} = s_C|\overline{E}^*, \Theta_{\overline{E}^*})P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi)d\Theta_{\overline{E}^*} \end{aligned} \tag{B.44}$$

$P(\mathcal{V} = S, \mathcal{C} = s_C|\overline{E}^*, \Theta_{\overline{E}^*})$ is determined by the expansion of equation 3.1 taking into account the TAN structure.

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\overline{E}^*, \Theta_{\overline{E}^*}) = \theta_C(s_C)\ \theta_{\rho_{\overline{E}}|C}(s_{\rho_{\overline{E}}}, s_C) \prod_{u,v\in\overline{E}} \theta_{v|u,C}(s_v, s_u, s_C) \tag{B.45}$$

$P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi)$ can be expanded from equations 8.21, 8.22, 8.25, 8.29, 8.30 and 8.31 into

$$\begin{aligned} P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi) = & D(\theta_C(.); N'_C(.)) \\ & \times \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(., c); N'_{\rho_{\overline{E}},C}(., c)) \\ & \times \prod_{c\in C} \prod_{u,v\in\overline{E}} \prod_{i\in A_u} D(\theta_{v|u,C}(., i, c); N'_{v,u,C}(., i, c)) \end{aligned} \tag{B.46}$$

Now we need to calculate the integral in equation B.44. We define:

$$1_C^{S,s_C}(c) = \begin{cases} 1 & c = s_C \\ 0 & \text{otherwise} \end{cases} \tag{B.47}$$

$$1_{i,C}^{S,s_C}(k, c) = \begin{cases} 1 & k = s_i \wedge c = s_C \\ 0 & \text{otherwise} \end{cases} \tag{B.48}$$

$$1_{i,j,C}^{S,s_C}(k, l, c) = \begin{cases} 1 & k = s_i \wedge l = s_j \wedge c = s_C \\ 0 & \text{otherwise} \end{cases} \tag{B.49}$$

It is easy to see that:

$$\sum_{j\in A_v} 1_{v,u,C}^{S,s_C}(j, i, c) = 1_{u,C}^{S,s_C}(i, c) \tag{B.50}$$

$$\sum_{i\in A_u} 1_{u,C}^{S,s_C}(i, c) = 1_C^{S,s_C}(c) \tag{B.51}$$

$$\sum_{c\in C} 1_C^{S,s_C}(c) = 1 \tag{B.52}$$

We can use this notation to expand the product $P(\mathcal{V} = S, \mathcal{C} = s_C|\overline{E}^*, \Theta_{\overline{E}^*})P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi)$ by substituting equations B.45 and B.46 giving:

$$\begin{aligned}
&P(\mathcal{V} = S, \mathcal{C} = s_C|\overline{E}^*, \Theta_{\overline{E}^*})P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi) = \\
&\quad = D(\theta_C(.); N'_C(.)) \prod_{c\in C} \theta_C(c)^{1_C^{S,s_C}(c)} \\
&\quad \times \prod_{c\in C} \left[D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)) \prod_{i\in A_{\rho_{\overline{E}}}} \theta_{\rho_{\overline{E}}|C}(i,c)^{1^{S,s_C}_{\rho_{\overline{E}},C}(i,c)} \right] \\
&\quad \times \prod_{c\in C} \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \left[D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)) \prod_{j\in A_v} \theta_{v|u,C}(j,i,c)^{1^{S,s_C}_{v,u,C}(j,i,c)} \right]
\end{aligned} \tag{B.53}$$

By analyzing equation B.53 we can see that the integral in equation B.44 can be calculated by applying the result in equation B.4 three times. This gives:

$$\begin{aligned}
P(\mathcal{V} = S, \mathcal{C} = s_C|E, \xi) &= \frac{\Gamma(\sum_{c\in C} N'_C(c))}{\prod_{c\in C}\Gamma(N'_C(c))} \frac{\prod_{c\in C}\Gamma(N'_C(c) + 1^{S,s_C}_C(c))}{\Gamma(\sum_{c\in C} N'_C(c) + 1^{S,s_C}_C(c))} \\
&\times \prod_{c\in C} \left[\frac{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}} N'_{\rho_{\overline{E}},C}(i,c))}{\prod_{i\in A_{\rho_{\overline{E}}}}\Gamma(N'_{\rho_{\overline{E}},C}(i,c))} \frac{\prod_{i\in A_{\rho_{\overline{E}}}}\Gamma(N'_{\rho_{\overline{E}},C}(i,c) + 1^{S,s_C}_{\rho_{\overline{E}},C}(i,c))}{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}} N'_{\rho_{\overline{E}},C}(i,c) + 1^{S,s_C}_{\rho_{\overline{E}},C}(i,c))} \right] \\
&\times \prod_{c\in C} \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \left[\frac{\Gamma(\sum_{j\in A_v} N'_{v,u,C}(j,i,c))}{\prod_{j\in A_v}\Gamma(N'_{v,u,C}(j,i,c))} \frac{\prod_{i\in A_v}\Gamma(N'_{v,u,C}(j,i,c) + 1^{S,s_C}_{v,u,C}(j,i,c))}{\Gamma(\sum_{i\in A_v} N'_{v,u,C}(j,i,c) + 1^{S,s_C}_{v,u,C}(j,i,c))} \right]
\end{aligned} \tag{B.54}$$

This expression can be simplified by applying equations 8.26,8.27,8.28,B.50,B.51 and B.52 and reorganizing:

$$\begin{aligned}
P(\mathcal{V} = S, \mathcal{C} = s_C|E, \xi) &= \frac{\Gamma(N')}{\Gamma(N'+1)} \\
&\times \prod_{c\in C} \prod_{i\in A_{\rho_{\overline{E}}}} \frac{\Gamma(N'_{\rho_{\overline{E}},C}(i,c) + 1^{S,s_C}_{\rho_{\overline{E}},C}(i,c))}{\Gamma(N'_{\rho_{\overline{E}},C}(i,c))} \\
&\times \prod_{u,v\in\overline{E}} \prod_{c\in C} \prod_{i\in A_u} \left[\frac{\Gamma(N'_{u,C}(i,c))}{\Gamma(N'_{u,C}(i,c) + 1^{S,s_C}_{u,C}(i,c))} \prod_{i\in A_v} \frac{\Gamma(N'_{v,u,C}(j,i,c) + 1^{S,s_C}_{v,u,C}(j,i,c))}{\Gamma(N'_{v,u,C}(j,i,c))} \right]
\end{aligned} \tag{B.55}$$

Since the quotient $\frac{\Gamma(N'_*(*)+1_*^{S,s_C}(*))}{\Gamma(N'_*(*))}$ is $N'_*(*)$ if the condition expressed by the $1_*^{S,s_C}(*)$ is satisfied and 1 otherwise we have that:

$$\begin{aligned} P(\mathcal{V}=S,\mathcal{C}=s_C|E,\xi) &= \frac{1}{N'} \\ &\times N'_{\rho_{\overline{E}},C}(s_{\rho_{\overline{E}}},s_C) \\ &\times \prod_{u,v\in\overline{E}}\left[\frac{N'_{v,u,C}(s_v,s_u,s_C)}{N'_{u,C}(s_u,s_C)}\right] \end{aligned} \tag{B.56}$$

Defining h_0^{S,s_C} and $h_{u,v}^{S,s_C}$ as in equations 8.34 and 8.35 it is easy to see that multiplying and dividing in equation B.56 by the factor:

$$\prod_{v\in V-\{\rho_{\overline{E}}\}} N'_{v,C}(s_v,s_C) \tag{B.57}$$

and rearranging we get:

$$P(\mathcal{V}=S,\mathcal{C}=s_C|E,\xi) = h_0^{S,s_C}\ Z_\beta \prod_{u,v\in E}\left(h_{u,v}^{S,s_C}\right) \tag{B.58}$$

and the expression depends only on the undirected structure of the tree.

Adding over Tree Structures

Combining equations B.43, B.58 we get

$$P(\mathcal{V}=S,\mathcal{C}=s_C|E,\xi)P(E|\xi) = h_0^{S,s_C} \prod_{u,v\in E}\left(\beta_{u,v}h_{u,v}^{S,s_C}\right) \tag{B.59}$$

Calculating the summation over structures using the matrix tree theorem for decomposable distributions gives the desired result.

$$P(\mathcal{V}=S,\mathcal{C}=s_C|\xi) = h_0^{S,s_C}|Q(\boldsymbol{\beta}\ \mathbf{h}^{S,s_C})| \tag{B.60}$$

□

B.3.2 Learning under decomposable distributions over TANs

Given that $P(M|\xi)$ follows a decomposable distribution over TANs with hyper-parameters $\boldsymbol{\beta},\mathbf{N}'$ we want to calculate $P(M|\mathcal{D},\xi)$ where $\mathcal{D}$ is an i.i.d. dataset sampled from a TAN distribution. Using Bayes rule we get:

$$P(M|\mathcal{D},\xi) = P(\overline{E}^*,\Theta_{\overline{E}^*}|\mathcal{D},\xi) = \frac{P(\overline{E}^*,\Theta^*_{\overline{E}}|\xi)P(\mathcal{D}|\overline{E}^*,\Theta_{\overline{E}^*},\xi)}{Z_{\mathcal{D}}} \tag{B.61}$$

The prior $P(\overline{E}, \Theta_{\overline{E}}|\xi)$ is calculated combining equations 8.21,B.43, B.46 giving:

$$\begin{aligned} P(\Theta_{\overline{E}^*}\overline{E}^*|\xi) = \frac{1}{Z_\beta} & \prod_{u,v\in\overline{E}} \beta_{u,v} \\ & \times D(\theta_C(.); N'_C(.)) \\ & \times \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)) \\ & \times \prod_{c\in C} \prod_{u,v\in\overline{E}} \prod_{i\in A_u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)) \end{aligned} \tag{B.62}$$

$P(\mathcal{D}|\overline{E}^*, \Theta_{\overline{E}^*}, \xi)$ is the probability that the model generates the data in $\mathcal{D}$. Since $\mathcal{D}$ contains independent identically distributed observations, we have that

$$\begin{aligned} P(\mathcal{D}|\overline{E}^*, \Theta_{\overline{E}^*}, \xi) = & \prod_{c\in C} \theta_C(c)^{N_C(c)} \\ & \times \prod_{c\in C} \prod_{i\in A_{\rho_{\overline{E}}}} \theta_{\rho_{\overline{E}}|C}(i,c)^{N_{\rho_{\overline{E}},C}(i,c)} \\ & \times \prod_{c\in C} \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \prod_{j\in A_v} \theta_{v|u,C}(j,i,c)^{N_{v,u,C}(j,i,c)} \end{aligned} \tag{B.63}$$

Substituting equations B.62 and B.63 into B.61 we get

$$\begin{aligned} P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) = & \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}} \prod_{u,v\in\overline{E}} \beta_{u,v} \\ & \times D(\theta_C(.); N'_C(.)) \prod_{c\in C} \theta_C(c)^{N_C(c)} \\ & \times \prod_{c\in C} \left[D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)) \prod_{i\in A_{\rho_{\overline{E}}}} \theta_{\rho_{\overline{E}}|C}(i,c)^{N_{\rho_{\overline{E}},C}(i,c)} \right] \\ & \times \prod_{c\in C} \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \left[D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)) \prod_{j\in A_v} \theta_{v|u,C}(j,i,c)^{N_{v,u,C}(j,i,c)} \right] \end{aligned} \tag{B.64}$$

Applying the result in equation B.3 for all the Dirichlets we have that

$$
\begin{aligned}
P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D},\xi) = & \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}}\prod_{u,v\in\overline{E}}\beta_{u,v} \\
& \times \frac{\Gamma(\sum_{c\in C} N'_C(c))}{\prod_{c\in C}\Gamma(N'_C(c))}\frac{\prod_{c\in C}\Gamma(N'_C(c)+N_C(c))}{\Gamma(\sum_{c\in C} N'_C(c)+N_C(c))} \\
& \times \prod_{c\in C}\left[\frac{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}} N'_{\rho_{\overline{E}},C}(i,c))}{\prod_{i\in A_{\rho_{\overline{E}}}}\Gamma(N'_{\rho_{\overline{E}},C}(i,c))}\frac{\prod_{i\in A_{\rho_{\overline{E}}}}\Gamma(N'_{\rho_{\overline{E}},C}(i,c)+N_{\rho_{\overline{E}},C}(i,c))}{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}} N'_{\rho_{\overline{E}},C}(i,c)+N_{\rho_{\overline{E}},C}(i,c))}\right] \\
& \times \prod_{c\in C}\prod_{u,v\in\overline{E}}\prod_{i\in A_u}\left[\frac{\Gamma(\sum_{j\in A_v} N'_{v,u,C}(j,i,c))}{\prod_{j\in A_v}\Gamma(N'_{v,u,C}(j,i,c))}\frac{\prod_{j\in A_v}\Gamma(N'_{v,u,C}(j,i,c)+N_{v,u,C}(j,i,c))}{\Gamma(\sum_{j\in A_v} N'_{v,u,C}(j,i,c)+N_{v,u,C}(j,i,c))}\right] \\
& \times D(\theta_C(.); N'_C(.)+N_C(.)) \\
& \times \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)+N_{\rho_{\overline{E}},C}(.,c)) \\
& \times \prod_{c\in C}\prod_{u,v\in\overline{E}}\prod_{i\in A_u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)+N_{v,u,C}(.,i,c))
\end{aligned}
\tag{B.65}
$$

This expression can be simplified by applying equations 8.26,8.27 and 8.28 (and similar ones for N) and reorganizing:

$$
\begin{aligned}
P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D},\xi) = & \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}}\frac{\Gamma(N')}{\Gamma(N'+N)}\prod_{u,v\in\overline{E}}\beta_{u,v} \\
& \times \prod_{c\in C}\prod_{i\in A_{\rho_{\overline{E}}}}\frac{\Gamma(N'_{\rho_{\overline{E}},C}(i,c)+N_{\rho_{\overline{E}},C}(i,c))}{\Gamma(N'_{\rho_{\overline{E}},C}(i,c))} \\
& \times \prod_{u,v\in\overline{E}}\prod_{c\in C}\prod_{i\in A_u}\left[\frac{\Gamma(N'_{u,C}(i,c))}{\Gamma(N'_{u,C}(i,c)+N_{u,C}(i,c))}\prod_{j\in A_v}\frac{\Gamma(N'_{v,u,C}(j,i,c)+N_{v,u,C}(j,i,c))}{\Gamma(N'_{v,u,C}(j,i,c))}\right] \\
& \times D(\theta_C(.); N'_C(.)+N_C(.)) \\
& \times \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)+N_{\rho_{\overline{E}},C}(.,c)) \\
& \times \prod_{c\in C}\prod_{u,v\in\overline{E}}\prod_{i\in A_u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)+N_{v,u,C}(.,i,c))
\end{aligned}
\tag{B.66}
$$

Defining $W_{u,v}$ as appears in equation 8.38, it is easy to see that multiplying and dividing in equation B.66 by the factor:

$$\prod_{v\in V-\{\rho_{\overline{E}}\}}\prod_{c\in C}\prod_{i\in A_v}\frac{\Gamma(N'_{v,C}(i,c)+N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} \tag{B.67}$$

and rearranging we get:

$$\begin{aligned}P(\overline{E}^*,\Theta_{\overline{E}^*}|\mathcal{D},\xi) &= \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}}\frac{\Gamma(N')}{\Gamma(N'+N)}\\ &\times\prod_{c\in C}\prod_{v\in V}\prod_{i\in A_v}\frac{\Gamma(N'_{v,C}(i,c)+N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))}\\ &\times\prod_{u,v\in\overline{E}}W_{u,v}\beta_{u,v}\\ &\times D(\theta_C(.);N'_C(.)+N_C(.))\\ &\times\prod_{c\in C}D(\theta_{\rho_{\overline{E}}|C}(.,c);N'_{\rho_{\overline{E}},C}(.,c)+N_{\rho_{\overline{E}},C}(.,c))\\ &\times\prod_{c\in C}\prod_{u,v\in\overline{E}}\prod_{i\in A_u}D(\theta_{v|u,C}(.,i,c);N'_{v,u,C}(.,i,c)+N_{v,u,C}(.,i,c))\end{aligned} \tag{B.68}$$

In order to have $P(\overline{E}^*,\Theta_{\overline{E}^*}|\mathcal{D},\xi)$ completely determined we need to calculate $Z_\mathcal{D}$. Since we know that

$$\int_{M\in\mathcal{M}}P(M|\mathcal{D},\xi)=\sum_{E\in\mathcal{E}}\int\cdots\int_{\Theta_{\overline{E}^*}}P(\overline{E}^*,\Theta_{\overline{E}^*}|\mathcal{D},\xi)=1 \tag{B.69}$$

We can do this by integrating over the parameters, then summing over the tree structures and finally solving for $Z_\mathcal{D}$. The first step is easy, because Dirichlet distributions are normalized and integrate to 1 giving:

$$\begin{aligned}\int\cdots\int_{\Theta_{\overline{E}^*}}P(\overline{E}^*,\Theta_{\overline{E}^*}|\mathcal{D},\xi) &= \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}}\frac{\Gamma(N')}{\Gamma(N'+N)}\\ &\times\prod_{c\in C}\prod_{v\in V}\prod_{i\in A_v}\frac{\Gamma(N'_{v,C}(i,c)+N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))}\\ &\times\prod_{u,v\in\overline{E}}W_{u,v}\beta_{u,v}\end{aligned} \tag{B.70}$$

The addition over structures can be calculated by means of the matrix tree theorem for decomposable priors, giving

$$\begin{aligned}\sum_{E\in\mathcal{E}}\int\cdots\int_{\Theta_{\overline{E}^*}}P(\overline{E}^*,\Theta_{\overline{E}^*}|\mathcal{D},\xi) &= \frac{|Q(\boldsymbol{\beta}\mathbf{W})|}{Z_\beta}\frac{1}{Z_\mathcal{D}}\frac{\Gamma(N')}{\Gamma(N'+N)}\\ &\times\prod_{c\in C}\prod_{v\in V}\prod_{i\in A_v}\frac{\Gamma(N'_{v,C}(i,c)+N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))}=1\end{aligned} \tag{B.71}$$

Solving for $Z_{\mathcal{D}}$, recalling that $Z_\beta = |Q(\boldsymbol{\beta})|$ we have that

$$Z_{\mathcal{D}} = \frac{|Q(\boldsymbol{\beta}\mathbf{W})|}{|Q(\boldsymbol{\beta})|} \frac{\Gamma(N')}{\Gamma(N'+N)} \prod_{c\in C} \prod_{v\in V} \prod_{i\in A_v} \frac{\Gamma(N'_{v,C}(i,c) + N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} \tag{B.72}$$

Finally, substituting the result for $Z_{\mathcal{D}}$ in equation B.68 we can see that the posterior is a decomposable distribution with the parameters updated as given by equations 8.36, 8.37 and 8.38:

$$\begin{aligned} P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D},\xi) = {} & \frac{1}{|Q(\boldsymbol{\beta}\mathbf{W})|} \prod_{u,v\in\overline{E}} W_{u,v}\beta_{u,v} \\ & \times D(\theta_C(.); N'_C(.) + N_C(.)) \\ & \times \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c) + N_{\rho_{\overline{E}},C}(.,c)) \\ & \times \prod_{c\in C} \prod_{u,v\in\overline{E}} \prod_{i\in A_u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c) + N_{v,u,C}(.,i,c)) \end{aligned} \tag{B.73}$$

□

Bibliography

Aczel, J. (1966). *Lectures on Functional Equations and their Applications*. Academic Press, New York.

Aha, D., Kibler, D., and Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, 6:37–66.

Andersson, S., Madigan, D., and Perlman, M. (1995). A characterization of markov equivalence classes for acyclic digraphs. Technical Report 287, Department of Statistics, University of Washington.

Arnborg, S. and Sjödin, G. (1999). On the foundations of bayesianism. Technical report, Nada, KTH.

Arnborg, S. and Sjödin, G. (2000a). Bayes rules in finite models. In *Proceedings of the European Conference on Artificial Intelligence*.

Arnborg, S. and Sjödin, G. (2000b). A note on the foundations of bayesianism. Technical report, Nada, KTH.

Bernardo, J. (1998). Bayesian reference analysis. a postgraduate tutorial course. Universitat de València, http://www.unine.ch/statistics/postgrad/images/Course.pdf.

Bernardo, J. (2003). Bayesian statistics. In *Encyclopedia of Life Support Systems (EOLSS)*.

Bernardo, J. and Ramón, J. (1998). An introduction to bayesian reference analysis. *The Statistician*, 47:101–135.

Blake, C., Keogh, E., and Merz, C. (1998). UCI repository of machine learning databases.

Catlett, J. (1991). On Changing Continuous Attributes into Ordered Discrete Attributes. In Kodratoff, Y., editor, *Proceedings of the European Working Session on Learning*, pages 164–178. Springer-Verlag.

Cerquides, J. (1997). Mantaras Distance for Discretization. Proposal and Empirical Comparison of a New Parallelizable Discretization Method. long version. Technical report, IIIA-97-03.

Cerquides, J. (1999a). Applying General Bayesian Techniques to Improve TAN Induction. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining, KDD99*, pages 292–296.

Cerquides, J. (1999b). Applying General Bayesian Techniques to Improve TAN Induction. Long Version. Technical report, Ubilab, http://www.ubilab.org.

Cerquides, J. and López de Màntaras, R. (1997a). Fuzzy metaqueries for guiding the Discovery Process in KDD. In *Proceedings of the Sixth International Conference on Fuzzy Systems*, volume Volume III, pages 1555–1559.

Cerquides, J. and López de Màntaras, R. (1997b). Proposal and empirical comparison of a parallelizable distance-based discretization method. In Heckerman, D., Mannila, H., Pregibon, D., and Uthurusamy, R., editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*.

Cerquides, J. and López de Màntaras, R. (1998a). A first analysis of qualitative influences and synergies. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence, AAAI-98 Student Abstracts Session*.

Cerquides, J. and López de Màntaras, R. (1998b). Knowledge discovery with qualitative influences and synergies. In *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD-98)*, volume 1510 of *LNAI*, pages 273–281. Springer.

Cerquides, J. and López de Màntaras, R. (1998c). A New Approach to Rule Interest Measures. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence, AAAI-98 Student Abstracts Session*.

Cerquides, J. and López de Màntaras, R. (1998d). A New Approach to Rule Interest Measures. long version. Technical report, IIIA-98-08.

Cerquides, J. and López de Màntaras, R. (1998e). Qualitative Influences and Synergies: A Step Forward in Explaining Probabilistic Reasoning. Long Version. Technical report, IIIA-98-04.

Cerquides, J. and López de Màntaras, R. (2003a). The indifferent naive bayes classifier. In *Proceedings of the 16th International FLAIRS Conference*, pages 341–345.

Cerquides, J. and López de Màntaras, R. (2003b). The indifferent naive bayes classifier. long version. Technical Report IIIA-2003-01, Institut d'Investigació en Intel.ligència Artificial.

Cerquides, J. and López de Màntaras, R. (2003c). Maximum a posteriori tree augmented naive bayes classifiers. Technical Report IIIA-2003-10, Institut d'Investigació en Intel.ligència Artificial.

Cerquides, J. and López de Màntaras, R. (2003d). Tractable bayesian learning of tree augmented naive bayes classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 75–82.

Cerquides, J. and López de Màntaras, R. (2003e). Tractable bayesian learning of tree augmented naive bayes classifiers. long version. Technical Report IIIA-2003-04, Institut d'Investigació en Intel.ligència Artificial.

Cestnik, B. (1990). Estimating probabilities: A crucial task in machine learning. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 147–149.

Chow, C. and Liu, C. (1968). Aproximatting Discrete Probability Distributions with Dependence Trees. *IEEE Transactions on Information Theory*, 14:462–467.

Clark, P. and Boswell, R. (1991). Rule induction with cn2: Some recent improvements. In Kodratoff, Y., editor, *Machine Learning - EWSL-91*, pages 151–163. Springer-Verlag.

Clark, P. and Niblett, T. (1989). The cn2 induction algorithm. *Machine Learning*, 3(4):261–283.

Cowell, R., Dawid, A., Lauritzen, S., and Spiegelhalter, D. (1999). *Probabilistic Networks and Expert Systems*. Springer-Verlag.

Cox, R. (1961). *The Algebra of Probable Inference*. John Hopkins University Press, Baltimore MD.

Dash, D. and Cooper, G. (2002). Exact model averaging with naive bayesian classifiers. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 91–98.

Dawid, A. (1983). Invariant prior distributions. In Kotz, S. and Johnson, N., editors, *Encyclopedia of Statistical Sciences*, volume 4, pages 228–236. Wiley.

Dawid, A. and Lauritzen, S. (1993). Hyper markov laws in the statistical analysis of decomposable graphical models. *The Annals of Statistics*, 21(3):1272–1317.

Devroye, L., Györfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. Springer Verlag.

Domingos, P. (1997). Bayesian model averaging in rule induction. In *Preliminary papers of the Sixth International Workshop on Artificial Intelligence and Statistics*, pages 157–164.

Dougherty, J., Kohavi, R., and Sahami, M. (1995). Supervised and Unsupervised Discretization of Continuous Features. In Prieditis, A. and Rusell, S., editors, *Machine Learning: Proceedings of the Twelfth International Conference*.

Elsaesser, C. (1989). Explanation of probabilistic inference. In Kanal, L., Levitt, T., and Lemmer, J., editors, *Uncertainty in Artificial Intelligence 3*, pages 387–400. Elsevier Science Publishers B.V. (North-Holland).

Eppstein, D. (1992). Finding the k smallest spanning trees. *BIT*, 32(2):237–248. Special issue for 2nd SWAT.

Fawcett, T. (2003). Roc graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, HP Laboratories Palo Alto.

Fayyad, U. M. and Irani, K. B. (1992). On the Hadling of Continuous-Valued Attributes in Decision Tree Generation. *Machine Learning*, 8:87–102.

Fayyad, U. M. and Irani, K. B. (1993). Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *13th International Joint Conference of Artificial Intelligence*, pages 1022–1027.

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29:131–163.

Friedman, N. and Goldszmidt, M. (1996). Discretizing Continuous Attributes While Learning Bayesian Networks. In *Internation Conference on Machine Learning*.

Gabow, H. (1977). Two algorithms for generating weighted spanning trees in order. *SIAM J. COMPUT.*, 6(1):139–150.

Gibbons, J. (1971). *Nonparametric statistical inference*. Series in probability and statistics. McGraw-Hill, New York.

Halpern, J. (1999a). A counterexample to theorems of cox and fine. *Journal of AI research*, 10:67–85.

Halpern, J. (1999b). Technical addendum, cox's theorem revisited. *Journal of AI research*, 11:429–435.

Hamming, R. (1980). *Coding and information theory*. Prentice-Hall, Englewood Cliffs (N.J).

Hanson, R., Stutz, J., and Cheeseman, P. (1991). Bayesian classification theory. Technical Report FIA-90-12-7-01, NASA Ames Research Center, Artificial Intelligence Branch.

Hartigan, J. (1964). Invariant prior distributions. *Ann. Math. Statist.*, 35:836–845.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer–Verlag.

Heckerman, D., Geiger, D., and Chickering, D. (1995). Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243.

Hoeting, J., Madigan, D., Raftery, A., and Volinsky, C. (1998). Bayesian model averaging. Technical Report 9814, Department of Statistics. Colorado State University.

Hulten, G. and Domingos, P. (2002). Mining complex models from arbitrarily large databases in constant time. In *Proc. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Jaynes, E. (1968). Prior probabilities. *IEEE Transactions on Systems Science and Cybernetics*.

Jaynes, E. (1988). How does the brain do plausible reasoning? In Erickson, G. and Smith, C., editors, *Maximum-Entropy and Bayesian Methods in Science and Engineering*, volume 1, pages 1–24. Kluwer Academic Publishers.

Jaynes, E. (1996). *Probability Theory: The Logic of Science*. published on the net, http://bayes.wustl.edu/Jaynes.book.

Johnson, E. (1973). Numerical encoding of qualitative expressions of uncertainty. Technical Report 250, Army Research Institute for the Behavioural and Social Sciences, Arlington, Virginia.

Kass, R. and Wasserman, L. (1994). Formal rules for selecting prior distribution: A review and annotated bibliography. Technical Report 583, Department of Statistics, Carnegie Mellon University.

Katoh, N., Ibaraki, T., and Mine, H. (1981). An algorithm for finding k minimum spanning trees. *SIAM J. Comput.*, 10(2):247–255.

Kerber, R. (1992). ChiMerge: Discretization of Numeric Attributes. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 123–128. MIT Press.

Keynes, J. (1921). *A Treatise on Probability*. MacMillan, London.

Kohavi, R., Becker, B., and Sommerfield, D. (1997). Improving simple bayes. In *Proceding of the European Conference in Machine Learning*.

Kohavi, R., John, G., Long, R., Manley, D., and Pfleger, K. (1994). MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence*, pages 740–743. IEEE Computer Society Press.

Kontkanen, P., Myllymaki, P., Silander, T., and Tirri, H. (1998). Bayes Optimal Instance-Based Learning. In *Machine Learning: ECML-98, Proceedings of the 10th European Conference*, volume 1398 of *Lecture Notes in Artificial Intelligence*, pages 77–88. Springer-Verlag.

Langley, P., Iba, W., and Thompson, K. (1992). An Analysis of Bayesian Classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 223–228. AAAI Press and MIT Press.

Lee, C. and Shin, D.-G. (1994). A Context-Sensitive Discretization of Numeric Attributes for Classification Learning. In *11th European Conference on Artificial Intelligence*, pages 428–432.

Lichtenstein, S. and Newman, J. (1967). Empirical scaling of common verbal phrases associated with numerical probabilities. *Psychon. Sci.*, 9(10).

López de Màntaras, R. (1991). A Distance Based Attribute Selection Measure for Decision Tree Induction. *Machine Learning*, 6:81–92.

López de Màntaras, R., Cerquides, J., and Garcia, P. (1998). Comparing Information-Theoretic Attribute Selection Measures: A Statistical Approach. *AI Communications*, 11(2).

MacKay, D. (1995). Bayesian methods for neural networks: Theory and applications. Neural Networks Summer School.

Madigan, D. and Raftery, A. (1994). Model selection and accounting for model uncertainty in graphical models using occam's window. *J. American Statistical Association*, 89:1535–1549.

Meila, M. and Jaakkola, T. (2000a). Tractable bayesian learning of tree belief networks. Technical Report CMU-RI-TR-00-15, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Meila, M. and Jaakkola, T. (2000b). Tractable bayesian learning of tree belief networks. In *Proc. of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 380 – 388.

Meila, M. and Jordan, M. I. (2000). Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48.

Neufeld, E. (1990). A probabilistic commonsense reasoner. *International Journal of Intelligent Systems*, 5:565–594.

Oden, G. (1977). Integration of fuzzy logical information. *Journal of Experimental Psychology: Human Perception and Performance*, 3(4):565–575.

Parsons, S. (1995). Further results in qualitative uncertainty. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 3(2):187–210.

Pazzani, M. (1995). An iterative improvement approach for the discretization of numeric attributes in bayesian classifiers. In *1st International Conference on Knowledge Discovery in Databases*.

Pettie, S. and Ramachandran, V. (2002). An optimal minimum spanning tree algorithm. *Journal of the ACM (JACM)*, 49(1):16–34.

Polya, G. (1954). *Mathemathics and Plausible Reasoning, Vol II: Patterns of Plausible Inference*. Princenton, New Jersey: Princenton University Press.

Provost, F., Fawcett, T., and Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Quinlan, J. (1992). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Richeldi, M. and Rossotto, M. (1995). Class-Driven Statistical Discretization of Continuous Attributes (Extendend Abstract). In *European Conference on Machine Learning*.

Rubey, M. (2000). Counting spanning trees. Diplomarbeit.

Shen, W. (1993). Bayesian probability theory. a general method for machine learning. Technical Report MCC-Carnot-101-93, Microelectornics and Computer Technology Corporation, Austin, TX.

Shoup, V. (2003). NTL: A library for doing number theory. http://www.shoup.net/ntl.

Thearling, K. (1998). Some thoughts on the current state of data mining software applications. In *Keys to the Commercial Success of Data Mining, KDD'98 Workshop*.

Trave, L. and Piera, N. (1989). The orders of magnitude models as qualitative algebras. In *11th IJCAI*, Detroit.

Tribus, M. (1969). *Rational Descriptions, Decisions and Designs*. Pergamon Press, New York.

Wallsten, T., Budescu, D., Rapoport, A., Zwick, R., and Forsyth, B. (1985). Measuring the vague meanings of probability terms. Technical Report 173, The L. L. Thurstone Psychometrich Laboratory, Chapel Hill, N.C.

Wellman, M. P. (1990). Fundamental concepts of qualitative networks. *Artificial Intelligence*, 44:257–303.

West, D. (1999). *Introduction to Graph Theory, Second Edition*. Prentice Hall.

Zimmer, A. (1983). Verbal vs. numerical processing of subjective probabilities. In Scholz, R., editor, *Decision Making Under Uncertainty*, pages 159–182. Elsevier Science Publishers B.V. (North-Holland).

Zimmer, A. (1985). The estimation of subjective probabilities via categorical judgments of uncertainty. In *Proceedings of the Workshop on Uncertainty and Probability in Artificial Intelligence*, pages 217–224. UCLA.