Motif representation and discovery

A.M. Carvalho

UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Motif representation and discovery

Alexandra Sofia Martins de Carvalho

Dissertação para obtenção do Grau de Doutor em
Engenharia Informática e de Computadores

Orientador:	Doutor Arlindo Manuel Limede de Oliveira
Co-orientadora:	Doutora Marie-France Sagot

Júri:

Presidente:	Presidente do Conselho Científico do IST
Vogais:	Doutor Paulo Jorge dos Santos Gonçalves Ferreira
	Doutor Arlindo Manuel Limede de Oliveira
	Doutor Mário Alexandre Teles de Figueiredo
	Doutora Marie-France Sagot
	Doutor João Manuel Portela da Gama
	Doutora Ana Teresa Correia de Freitas

Julho de 2011

Abstract

An important part of gene regulation is mediated by specific proteins, called *transcription factors*, which influence the transcription of a particular gene by binding to specific sites on DNA sequences, called *transcription factor binding sites* (TFBS) or, simply, *motifs*. Such binding sites are relatively short segments of DNA, normally 5 to 25 nucleotides long, over-represented in a set of co-regulated DNA sequences. There are two different problems in this setup: *motif representation*, accounting for the model that describes the TFBS's; and *motif discovery*, focusing in unravelling TFBS's from a set of co-regulated DNA sequences.

This thesis proposes a discriminative scoring criterion that culminates in a discriminative mixture of Bayesian networks to distinguish TFBS's from the background DNA. This new probabilistic model supports further evidence in non-additivity among binding site positions, providing a superior discriminative power in TFBS's detection. On the other hand, extra knowledge carefully selected from the literature was incorporated in TFBS discovery in order to capture a variety of characteristics of the TFBS's patterns. This extra knowledge was combined during the process of motif discovery leading to results that are considerably more accurate than those achieved by methods that rely in the DNA sequence alone.

Keywords: Motif representation, Discriminative learning, Bayesian network, Motif discovery, Combinatorial algorithm, Position specific prior.

Resumo

Uma parte importante da regulação genética é mediada por proteínas específicas, chamadas *factores de transcrição*, que influenciam a transcrição de um gene através da ligação a locais específicos das sequências de ADN, denominados *locais de ligação dos factores de transcrição* (LLFT) ou, simplesmente, *motivos.* Estes locais de ligação são pequenos segmentos de ADN, estendendo-se normalmente de 5 a 25 bases, que se encontram sobrerrepresentados num conjunto de sequências de ADN co-reguladas. Consideram-se dois problemas: *representação de motivos*, que descreve o modelo dos LLFT's; e *descoberta de motivos*, que se foca na descoberta de LLFT's a partir de um conjunto de sequências de ADN co-reguladas.

Esta dissertação propõe uma função de custo discriminativa que culmina numa mistura de redes de Bayes para distinguir os LLFT's do restante ADN. Este novo modelo probabilístico evidência a existência de dependência nas posições dos locais de ligação, oferecendo um poder superior de discriminação de LLFT's. Por outro lado, informação adicional seleccionada da literatura foi incorporada na descoberta de motivos por forma a capturar uma variedade de características dos padrões de LLFT's. Esta informação extra promoveu uma detecção de motivos consideravelmente mais correcta quando comparada com métodos que se baseiam apenas na sequência de ADN.

Palavras chave: Representação de motivos, Aprendizagem discriminativa, Rede de Bayes, Descoberta de motivos, Algoritmo combinatório, *Prior* para posição específica.

To Duarte and Tomás,

Acknowlegements

I am heartily thankful to my supervisors, Arlindo Oliveira and Marie-France Sagot, for useful scientific discussions and their guidance during the elaboration of this thesis. I would like to thank Arlindo for always having a slot in his heavy agenda to discuss my work, and for his talent in giving me support while coping with my working peaks dictated by two maternities and teaching as a teaching Assistant at IST. To Marie-France, I thank her hospitality during my short visits to the BAOBAB research group and the opportunity she gave me to meet excellent scientists, namely, Nadia Pisanti, Laurent Marsan and Richard Lavery.

It is an honor for me to thank to all researchers I work with during the course of this thesis. In the first two years I was able to cooperate with Nadia and Laurent, jointly with Marie-France, I owe them enthusiastic discussions about RISOTTO. I also would like to show my gratitude to Teemu Roos and Petri Myllymäki, whom I have not met personally yet, for their eagerness and commitment in leading to a good port the work about $\hat{\text{f}}$CLL. I hope we met personally someday.

Additionally, I offer my regards to Mário Figueiredo, for his availability in helping me with concerns that appeared with respect to the $\hat{\text{f}}$CLL work, and to Víctor Rocha Vieira, for his interest in cross-checking the analytical integrations presented in this work (I still have the hand-made 6-page integral calculations). Finally, I would like to thank PRIORITY researchers, namely, Raluca Gordân, and MEME researchers, namely, Timothy Bailey, for making available data from their works, making possible the contribution of GRISOTTO.

I would like to thank the friendship and support that some of my colleagues at IST demonstrated about the completion of this thesis: Luís Miguel Silveira, for his concern and belief; Rui Gustavo Crespo, for asking for the “3 magic letters” so many times for so many years; João Cunha Serra, for avidly helping me in so many doubts about academic career. Last, but not least, I would like to thank Cristina and Amílcar Sernadas for our conversations that, albeit indirectly, made me believe that this though journey was more or less normal.

I am profoundly grateful to the excellent working environment that I have in KDBIO Group that houses me at INESC-ID. I owe a word of gratitude to all my colleagues, specially: Ana Teresa Freitas, for always caring how things were, including me, and for supporting my work when I was aboard; Susana Vinga, a great friend, for all her patience and uplifting words about my work; and Sara Madeira, for all conversations about academic life and life in general. During these years, I had outstanding office mates with whom I had several discussions about work and life in general, including, Orlando Anunciação, Miguel Bugalho, Alexandre Francisco, Joana Gonçalves, Luís Russo, Pedro Monteiro and Nuno Mendes. I also offer my regards to Sara Silva, Paulo Fonseca, João Carriço and Cátia Vaz for our conversations about gaming, kids, life in general, and academic career, to name a few.

Lastly, I offer my regards and blessings to my family and friends for all support and patience in any respect during the completion of this thesis. *Aos meus pais, Bernandina e Francisco, por terem apostado em mim desde sempre, um grande xi-prata para vocês. À Irene e ao Agostinho não poderia deixar de agradecer com um muito obrigado, sem a vossa ajuda nunca teria conseguido. À minha irmã, Filipa, sempre com um fácil e lindo sorriso, e ao Faustino, um muito obrigado por todo o apoio, eu e os meus filhos adoramos-vos. À Ritinha, a mais nova da família, bem-vinda! Ao Pedro e à Sofia, e à Eva e ao Leonardo, um muito obrigado pelos momentos descontraídos que passámos juntos. À minha tia Almerinda e Adelaide, um muito obrigado por várias vezes me terem ajudado com a criançada.* To Sara, a huge thanks for hosting me in your house in my less fortunate moments, it was great! To Susana, for listening me all days, over and over, and for always having the right words to cheer me up. Finally, to Paulo that has been a pillar of support I could rely on all times. He inspired me, encouraged me, and sustained me. Most of all, he has always believed in me. *E aos nosso lindos filhotes, Duarte e Tomás, que são o sol da minha vida, um muito obrigado pelos inúmeros e calorosos abraços que tanto gosto.*

Contents

I Background **1**

1 Introduction **3**

1.1 Context . . . 3

1.2 Aims . . . 4

1.3 Claim of contributions . . . 4

1.4 Layout of the thesis . . . 7

2 State of the art **9**

2.1 Problem description . . . 9

2.2 Motif representation . . . 12

2.2.1 Deterministic models . . . 12

2.2.2 Probabilistic models . . . 12

2.3 Motif discovery . . . 16

3 Related Work **21**

3.1 Bayesian network models . . . 21

3.1.1 Bayesian networks . . . 21

3.1.2 Bayesian network classifiers . . . 33

3.2 Extraction of structured motifs . . . 37

3.2.1 Basic data structures . . . 38

3.2.2 Single motif extraction algorithm . . . 41

3.2.3 Structured motif extraction algorithm . . . 45

II Motif representation **53**

4 $\hat{\text{f}}$CLL: Factorized conditional log-likelihood **55**

4.1 Notation 59

4.2 Generative vs discriminative learning 60

4.3 Developing a new scoring criterion 62

4.3.1 Achieving a well-behaved approximation under OFE 72

4.3.2 Information-theoretical interpretation 78

4.3.3 Beyond binary classification 81

4.4 Experimental results 82

5 CκG: Learning consistent κ-graphs **93**

5.1 Biological motivation for CκG Bayesian networks 93

5.2 Issues and trends in efficient learning of Bayesian networks 94

5.3 Consistent Bayesian networks 96

5.3.1 CκG classifier: An extension to the TAN classifier 101

5.3.2 Expressiveness of CκG Bayesian networks 102

5.3.3 Discriminative learning of two-component mixtures of CκG Bayesian networks 103

5.4 Experimental results 108

III Motif discovery **113**

6 RISOTTO: Improving RISO with maximal extensibility **115**

6.1 Single motif extraction 116

6.1.1 Using maximal extensibility of factors 116

6.1.2 Complexity analysis 120

6.2 Structured motif extraction 124

6.2.1 Using maximal extensibility of factors 125

6.3 Implementation and experimental results 126

6.3.1 Storing the extensibility information 127

6.3.2 Experimental results 128

7 GRISOTTO: Improving RISOTTO with prior knowledge 131

7.1 GRISOTTO algorithm . . . 132

7.2 Balanced information score . . . 135

7.3 Experimental results . . . 138

7.3.1 ChiP-chip data . . . 139

7.3.2 ChiP-seq data . . . 144

7.4 Discussion . . . 147

IV Conclusions and future work 151

V Appendixes 159

A Alternative justification for Assumption 1 161

B Feeding GRISOTTO with good initial starting points 163

B.1 RISOTTO input . . . 163

B.2 GRISOTTO subroutine calling RISOTTO . . . 164

C Inter-motif distance 167

C.1 Minimum scaled Euclidean distance . . . 168

C.2 PSSM representation of an IUPAC string . . . 168

C.3 Best alignment and cutoffs . . . 174

D Evaluating various positional priors 177

D.1 Parameter settings . . . 177

D.2 Running times . . . 179

D.3 Detailed results . . . 179

Bibliography 183

List of Algorithms

3.1 Chow-Liu tree learning algorithm, LL score 32

3.2 Learning tree BN's, decomposable and score equivalent ϕ–score 32

3.3 Learning tree BN's, decomposable and non-score equivalent ϕ–score 33

3.4 Learning TAN BNC's, LL score 36

3.5 Learning TAN BNC's, decomposable and score equivalent ϕ–score 37

3.6 Learning TAN BNC's, decomposable and non-score equivalent ϕ–score 37

3.7 SPELLER, single motif extraction 45

3.8 RISO, structured motif extraction using box-links 49

5.1 Learning CκG BN's, decomposable ϕ-score 99

5.2 Learning CκG BNC's, decomposable ϕ-score 101

5.3 Learning CκG mixture models for binary classification tasks 108

6.1 SPELLER, single motif extraction (simplified version) 116

6.2 RISOTTO, single motif extraction with maximal extensibility 121

6.3 RISO, structured motif extraction (simplified version) 125

6.4 RISOTTO, structured motif extraction with maximal extensibility 127

7.1 GRISOTTO, Greedy RISOTTO 135

7.2 GGP, GRISOTTO greedy procedure 136

B.1 RunRISOTTO, RISOTTO parameter tuning 165

C.1 ComputeDistance, minimum scaled Euclidean distance with cutoffs 175

List of Figures

3.1 Suffix tree for string AGACAGGAGGC\$. 39

3.2 Generalized suffix tree for strings TACTA\$ and CACTCA\$. 40

3.3 Generalized suffix tree with *Colors* for strings TACTA\$ and CACTCA\$. . . . 41

3.4 At most 5-deep factor tree for string AGACAGGAGGC\$. 42

3.5 At most 3-deep factor tree for string AGACAGGAGGC\$. 42

3.6 A general idea of box-links. 47

3.7 Extracting structured motifs following box-links. 49

4.1 Comparison between f (left) and $\hat{f}$ (right). 68

4.2 Approximation error between f and $\hat{f}$. 68

4.3 Plot of g and $\hat{g}$. 77

4.4 Scatter plots of the accuracy of Bayesian network-based classifiers. 88

4.5 Scatter plot of the accuracy of proposed methods against state-of-the-art classifiers. 89

5.1 Figure relative to the Example 5.3.3. 98

5.2 Expressiveness of the network models discussed in this work. 102

5.3 Scatter plots of the accuracy of different multinet classifiers. 111

6.1 Example where the extension of m' can be avoided. 118

6.2 Ratio between the expected number of extensions attempted by RISOTTO and RISO. 124

6.3 Ratio between the number of extensions attempted by RISOTTO and RISO. 129

6.4 Ratio between performance of RISOTTO and RISO. 130

List of Tables

2.1 The IUPAC nucleotide code with corresponding DNA bases. 14

4.1 Definition of terms used in Chapter 4. 61

4.2 Description of datasets used in the experiments. 83

4.3 Bayesian network-based classifiers used in the experiments. 84

4.4 Other state-of-the-art classifiers used in the experiments. 84

4.5 Accuracy of Bayesian network-based classifiers annotated with the standard deviation. 86

4.6 Accuracy of other state-of-the-art classifiers annotated with the standard deviation. 87

4.7 Statistical significance of the results achieved by the Bayesian network-based classifiers according to the Wilcoxon signed-rank test. 90

4.8 Statistical significance of the results achieved by the other state-of-the-art classifiers according to the Wilcoxon signed-rank test. 90

5.1 Definition of terms used in Chapter 5. 96

5.2 Definition of terms used in Section 5.3.3. 105

5.3 Statistical significance of the results according to the Wilcoxon signed-rank test. 109

6.1 Definition of terms used in Chapter 6. 117

7.1 Definition of terms used in Chapter 7. 134

7.2 Comparison of GRISOTTO with state-of-the-art methods over ChiP-chip data. 141

7.3 Comparison of GRISOTTO-$\mathcal{DC}$ with Chen *et al.* and MEME-$\mathcal{DC}$ over ChiP-seq data. 146

C.1 Canonical distribution of the IUPAC symbols 169
C.2 Distribution of the IUPAC symbols with three types of errors. 171
C.3 Translation of IUPAC symbols whose distance is closer to the average distance. 172
C.4 Average distance between PSSM's representing IUPAC symbols. 173

Part I

Background

Chapter 1

Introduction

1.1 Context

The core of the work presented in this thesis was developed at the Knowledge Discovery and Bioinformatics Group (KDBIO Group) of INESC-ID, Lisboa. The Bioinformatics effort in KDBIO Group aims at several goals. One of these goals is the focus of this thesis: the development of efficient algorithms for motif representation and discovery.

Some techniques used throughout this thesis rely and are greatly influenced by previous research works devised at the BAOBAB research group of INRIA Rhône-Alpes, leaded by Marie-France Sagot. Actually, one of the contributions of this thesis appeared as a collaborative work with two post-graduate students of Marie-France, namely, Nadia Pisanti and Laurent Marsan. There were other approaches that also influenced the work of this thesis, mainly, those of Raluca Gordân, Timothy Bailey, Nir Friedman and Russell Greiner. The latter was actually an intermediary for another collaborative work with Teemu Roos and Petri Myllymäki from the Helsinki Institute for Information Technology HIIT (U. Helsinki).

This work was partially supported by the PhD grant from FCT SFRH/BD/18660/2004, and by the projects: DBYeast[1] FCT Project POSC/EIA/57398/2004; ARN[2] FCT Project PTDC/EIA/67722/2006; Dyablo[3] FCT Project PTDC/EIA/71587/2006; and, PneumoSyS[4]

[1]DBYeast: Infrastructures and algorithms for analysis and identification of gene regulatory networks.

[2]ARN: Algorithms for the identification of genetic regulatory networks.

[3]Dyablo: Models for the dynamic behavior of biological networks.

[4]PneumoSyS: A systems biology approach to the role of pneumococcal carbon metabolism in colonization and invasive disease.

FCT Project PTDC/SAU-MII/100964/2008. This work was also supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds.

1.2 Aims

In this large-scale genome sequencing era the main bottleneck to progress in molecular biology is data analysis. The prime objective of this thesis is the investigation of one kind of biological information contained in sequenced data: the motif model and its discovery from a set of co-regulated DNA sequences. A motif is roughly a mathematical representation underlying a transcription factor binding site.

More precisely, the main goal of this thesis is the proposal of efficient and effective algorithms for motif representation and discovery, capable of dealing with the enormous amount of data coming from the Bioinformatics community. Such models and algorithms should be able to look in and beyond the DNA sequence alone, preferably gathering information from different sources. This grounds in the belief that such diverse information would increase the discriminative power of current tools. To its possible extent, the improvements obtained should be documented by complexity analysis, as well as by experimental results over biologically relevant sequence-sets.

1.3 Claim of contributions

Four main contributions were achieved within the scope of this thesis:

I. Motif representation

- A new efficient and parameter-free scoring criterion for learning augmented naive Bayes classifiers was devised (Carvalho, Roos, Oliveira, and Myllymäki, 2011). The new score, named *factorized conditional log-likelihood* ($\hat{\text{f}}$CLL), consists in an unbiased approximation to the *conditional log-likelihood* (CLL). The approximation was devised in order to guarantee decomposability over the network structure, as well as efficient estimation of the optimal parameters, achieving the same time and space complexity of the traditional log-likelihood scoring criterion. With this approach we achieved, although approximate, full discriminative learning of augmented naive

Bayes classifiers very efficiently. Full discriminative learning of Bayesian network classifiers is an open question in the Machine Learning community. Therefore, results concerning the $\hat{\text{f}}$CLL contribution were presented with benchmark datasets specially devised by that community. Notwithstanding, discriminative learning is extremely important in classification tasks, as the one of discriminating binding sites from background DNA sequences, and it was investigated in the course of this thesis for this very purpose. Actually, an application of $\hat{\text{f}}$CLL for motif representation is reported in the next contribution.

- A new probabilistic motif model accounting for non-additivity along positions in TFBS's, called *consistent κ-graph* (CκG) Bayesian networks, was proposed together with the machinery to learn it (Carvalho, Oliveira, and Sagot, 2007; Carvalho and Oliveira, 2007). Firstly, it was proposed a polynomial-time algorithm for learning CκG Bayesian networks (Carvalho and Oliveira, 2007). Afterwards this result was further extended to deal with classification tasks (Carvalho, Oliveira, and Sagot, 2007). In the course of this thesis we additionally exploited the aforementioned works by considering an extension of the discriminative scoring criterion, $\hat{\text{f}}$CLL, to two-component mixtures of CκG models, called *mixture-based factorized conditional log-likelihood* (m$\hat{\text{f}}$CLL). The rational for this approach is that there are two separate regimes underlying motifs and correspondent background DNA sequences, and mixtures allow discriminative learning of the motifs detached from the generative learning of the background.

II. Motif discovery

- An extension to RISO (Carvalho, Freitas, Oliveira, and Sagot, 2006, 2005; Carvalho, 2004) capable of dealing with long motifs was achieved with RISOTTO (Pisanti, Carvalho, Marsan, and Sagot, 2006) by using *maximal extensibility* information. In 2005, Nadia Pisanti, Laurent Marsan, and Marie-France Sagot, appeared with a proposal to improve the SPELLER motif extraction algorithm (Sagot, 1998) with maximal extensibility information.[5] At that time there were

[5]Despite the author of the thesis being listed as the second co-author of the paper, both the first and the second author worked equally for the paper.

several open questions: (1) how to use maximal extensibility for motifs with spacers, referred as *structured motifs*; (2) how to efficiently encode maximal extensibility in memory; (3) theoretical study concerning the average case complexity analysis of the maximal extensibility. The author of this thesis solved all these three issues, implemented the solution, designed and performed the experiments, and wrote the final paper together with Nadia.

- An extension to RISOTTO having the ability to take into consideration information in and beyond the DNA sequence alone was proposed (Carvalho and Oliveira, 2011). This extra information can be taken from the literature, or computed from the DNA sequences, and helps in characterizing motifs. The new algorithm, called GRISOTTO, combines and incorporates this extra information in a new theoretical-information scoring criterion, called *balanced information score* (BIS). Three available priors from different sources, namely, orthologous conservation, DNA duplex stability and nucleosome positioning, were combined into the BIS score. The GRISOTTO with the just mentioned BIS scoring criterion has shown to be the more accurate motif discoverer among twelve other state-of-the-art approaches for the same task.

Since this thesis started, the author has also made other contributions directly related with the work of this thesis. These works have not been included in the contributions of the thesis and are detailed below:

- Exposing RISO, developed by the author in her Master Thesis, internationally, in a top journal (Carvalho, Freitas, Oliveira, and Sagot, 2006), and international conference of the area (Carvalho, Freitas, Oliveira, and Sagot, 2005).

- Improving RISO with a double stranded feature which was included in the YEASTRACT-DISCOVERER (Monteiro, Mendes, Teixeira, d'Orey, Tenreiro, Mira, Pais, Francisco, Carvalho, Lourenço, Sá-Correia, Oliveira, and Freitas, 2008).

- Studying and comparing diverse scoring criteria for learning augmented naive Bayes classifiers from complete data (Carvalho, 2009).

Finally, the following software packages were developed and made available during the development of this thesis:

- A Java package extending WEKA (Hall et al., 2009) with the local-based score $\hat{\text{f}}$CLL to learn Bayesian networks classifiers can be found at
  `http://kdbio.inesc-id.pt/∼asmc/software/fCLL.html`

- A *Mathematica* package for learning CκG Bayesian networks and multinets for commonly used decomposable scores, including, $\hat{\text{f}}$CLL and m$\hat{\text{f}}$CLL, can be found at
  `http://kdbio.inesc-id.pt/∼asmc/software/CkG.html`

- An ansi-C implementation of RISO motif discovery algorithm from where RISOTTO was extended, can be found at
  `http://kdbio.inesc-id.pt/∼asmc/software/riso.html`,
  and also within the YEASTRACT-DISCOVERER at
  `http://www.yeastract.com/discoverer/riso.php`.

- An ansi-C implementation of RISOTTO motif discovery algorithm to deal with long motifs can be found at
  `http://kdbio.inesc-id.pt/∼asmc/software/risotto.html`

- A Java package with GRISOTTO motif discovery algorithm that uses prior information can be found at
  `http://kdbio.inesc-id.pt/∼asmc/software/grisotto.html`

1.4 Layout of the thesis

In Part I of the thesis, comprising Chapter 2–3, we introduce the background. In Chapter 2 we describe the problem focused in this thesis as well as the state-of-the-art providing an overview on motif representation and discovery. Some of the works mentioned in the state-of-the-art that are more closely related to the topics of the thesis are then presented in detail in Chapter 3. This chapter is divided in two main sections. First, Section 3.1 describes some background on learning Bayesian networks classifiers which plays an important role to understand Part II of the thesis that concerns motif representation. Second, Section 3.2 describes algorithms for motif discovery, namely, RISO (Carvalho, Freitas, Oliveira, and Sagot, 2006; Carvalho, 2004), which is used as background for Part III that addresses motif discovery.

The contributions of this thesis are described from Chapter 4 to Chapter 7. Part II refers to motif representation and comprises Chapter 4–5. In Chapter 4 we propose a new scoring criterion, called *factorized conditional log-likelihood* ($\hat{\text{f}}$CLL), for learning Bayesian networks devoted to classification tasks. In Chapter 5 we present a new motif model, called *consistent k-graphs* (CκG), which relies on a Bayesian network to introduce dependencies among motif sites. Part III of the thesis concentrates on motif discovery and includes Chapter 6–7. In Chapter 6 we present a new exact combinatorial motif discovery algorithm, called RISOTTO, to deal with long motifs. In Chapter 7 we present a new greedy approach, called GRISOTTO, to improve RISOTTO motif discoverer with prior knowledge.

Finally, in Part IV we draw some conclusions and propose future work and in Part V we provide four appendixes. Appendix A presents an alternative justification to an assumption presented in Chapter 4, whereas Appendix B–D concerns to the evaluation methodology and detailed results of the experimental results presented in Chapter 7.

Chapter 2

State of the art

This chapter starts by describing the problem focused in this thesis. Next, it provides the state-of-the-art on motif representation and discovery.

2.1 Problem description

The identification of DNA as the genetic material revealed that genetic information is represented by a sequence of four *bases* (A, C, G and T), also known as *nucleotides*. In molecular terms, a *gene* can be defined as a segment of DNA. Genes act by coding the structure of proteins, which are responsible for directing cell metabolism through their activity as enzymes. The central dogma of molecular biology assumes a pathway for the flow of genetic information: DNA $\rightarrow$ RNA $\rightarrow$ protein. According to this principle, RNA molecules are synthesized from DNA templates, a process called *transcription*, and proteins are synthesized from RNA templates, a process called *translation*. RNA appears therefore as an intermediate to convey information from DNA to the places of protein synthesis.

The complete genetic content, called *genome*, of most *eukaryotes* (cell or organism provided with a distinct nuclear envelop) is larger and more complex than the genetic content of *prokaryotes* (cell or organism that lack a nuclear envelope, also called *bacteria*). In fact, the genome of most eukaryotic organisms contains not only functional genes, but also large amounts of DNA sequences that do not code for proteins. Some of these non-coding DNA sequences lie between genes, in the so-called *intergenic regions*. However, large amounts of non-coding DNA are also found within the genes. Actually, genes of eukaryotic organisms are composed of segments of coding sequences, called *exons*, separated by segments of non-coding sequences, called *introns*.

An important part of gene regulation is mediated by specific proteins, called the *transcription factors* (TF), which influence the transcription of a particular gene by binding to specific sites on DNA sequences, called *transcription factor binding sites* (TFBS) or simply *binding sites*. Such binding sites are relatively short stretches of DNA and are located in the so-called *promoter regions*. These binding sites are short. The effective length may be just 4–6 nucleotides, although the region affected by the TF is longer, typically 10–25 nucleotides. Most of these regions are located in the non-coding sequences upstream of genes, but some are also found downstream, and even within the non-coding parts of a gene, the introns. In prokaryotic organisms, the binding sites are located predominantly in the immediate vicinity of the gene, which usually extends about 300 to 600 nucleotides upstream of the *transcription start site* (TSS). However, in eukaryotic organisms the binding site sequences are often shorter, and can be quite variable and distributed over very large distances. There is no clear-cut defined boundary for promoter regions which may extend further upstream to more than 2000 bases, as observed in some sea urchin promoters (Kirchhamer et al., 1996).

Promoter prediction necessarily needs a model of promoter organization and its conspicuous features. In fact, strong and weak points of promoter prediction methods are determined to a large extent by the accuracy of the underlying promoter model with respect to the biological organization. A possible way to describe a promoter views it as being composed of three regions with different functions, each one having one or more TFBS's. The first one, the *core promoter*, is the region that suffices to determine the precise TSS. The second one, the *proximal promoter*, is the region that is capable of initiating basal transcription. Finally, the *distal promoter*, also called *enhancer*, is the transcription regulatory region that can be located farther from the core promoter and has the main function of stimulating transcription. A detailed explanation on possible models for prediction and recognition of eukaryotic promoters was developed by Werner (1999).

The DNA sites involved in promoter function can be identified by searching for well conserved regions in a set of non-coding DNA sequences. Such well conserved regions, also known as consensus regions, are called *motifs* and can be found by comparison of non-coding sequences of a given organism, or by comparison of non-coding sequences of related genes in different organisms. In the first approach, frequently occurring patterns are likely to correspond to binding sites of a common TF. The second approach is called *phylogenetic footprinting* (Duret and Bucher, 1997) and requires careful identification of the appropriate genes to use. Only non-coding sequences of *orthologous genes*, which are genes evolutionarily related that perform the same biological function, are appropriate for phylogenetic footprinting. This technique uses the functional/non-functional dichotomy to identify regulatory elements by finding unusually well conserved regions in a set of orthologous non-coding DNA sequences from multiple species (for example, the non-coding sequence upstream the insulin gene in different species of vertebrates). Functional sequences tend to evolve much slower than non-functional ones, as they are subject to selective pressure. Hence, it is a good conjecture that unusually well conserved regions in such sequences have some regulatory function.

There are two central problems concerning motifs in sequences: localization and discovery (Crochemore and Sagot, 2004). The goal of the *motif localization* problem is to find the positions of the occurrences of a given motif in a DNA sequence (Policriti et al., 2004). The *motif discovery* problem, also called *motif extraction* problem, aims at identifying *de novo* binding site consensi from a set of non-coding DNA sequences. In both these problems, the accuracy of the underlying motif model is of the utmost importance. Indeed, an inaccurate model may lead to a high false positive rate in motif localization and discovery. This thesis focus in the representation and discovery tasks. Despite the existence of several proposals in the literature for motif discovery, the problem of detecting regulatory sites in DNA sequences is far from being solved. Given the flexibility of regulatory mechanisms, it remains essential to develop computer-assisted promoter recognition methods capable of detecting different kinds of regulatory signals and adapting to different promoter models. The impact of this task in the Bioinformatics community is enormous. Promoter regions can play an important role in gene function and may offer some clues to the function of completely anonymous proteins. Prediction of the functionality of a promoter may also yield initial indications for gene therapy approaches, while analysis of the combinatorics of their elements is essential for understanding cell development.

2.2 Motif representation

Herein we focus our attention on how to represent a motif from a collection of binding site sequences.

2.2.1 Deterministic models

There are two main kinds of deterministic models: regular expressions and consensus sequences. The *regular expressions* used in motif discovery denote a subset of regular languages and are typically composed of exact symbols, ambiguous symbols, fixed gaps and/or flexible gaps (Brazma et al., 1998a). A *consensus sequence* represents a collection (or neighborhood) of binding site sequences that are at most at a certain *Hamming distance*[1] of the underlying consensus sequence. Each binding site sequence in this collection is called a *motif occurrence* or *consensus occurrence*. The number of mismatches depends largely on the size of the motif. There are a few variants to these two models. A first alternative imposes a restriction on the location of mismatches along the consensus sequence (Pavesi et al., 2001). That is, a consensus occurrence can present at most a certain number of mismatches in the first i nucleotides, and so on. On the other hand, a second variant takes into account the sum of mismatches between all consensus occurrences and the underlying consensus sequence (Li and Fu, 2005).

2.2.2 Probabilistic models

The main drawback of deterministic models is that they lose some information, as compared with the collection of binding site sequences from where they are generated. For instance, even if we know that a consensus sequence has at least 2 mismatches within the collection of binding sites, we do not know if there are one or more bases which are specially well conserved, nor, for those bases that are not so well conserved, what kind of mismatches they have. The probabilistic models appeared to overcome such loss of information.

The *position specific scoring matrix* (PSSM), also known as *probability weight matrix* (PWM), is, without doubt, one of the most widely used probabilistic models. This model is represented by a matrix where each entry (i, b) is the probability of base b at the i-th position in the collection of binding sites. The information summarized in a PSSM can also

[1]The Hamming distance between two string measures the minimum number of substitutions required to change one string into the other.

be represented by a motif logo. The *motif logos* are based on the *information content* of the collection of binding sites. The information content at a position i in a site is defined as

$$I_i = 2 + \sum_{b=A}^{T} f_{b,i} \log_2 f_{b,i}$$

where b refers to the DNA bases and $f_{b,i}$ is the frequency of base b at that i-th position in the collection of binding sites. I_i is 0 for positions that are 25% of each base, and 2 for positions completely conserved. Bases are stacked on top of each other in increasing order of their frequencies and the size of each base printed in the logo is determined by multiplying the frequency of that base by the total information at that position, that is, $f_{b,i}I_i$.

Another quite popular motif model that is, in point of fact, the most widely used motif representation among biologists and TFBS's databases (e.g. Wingender et al., 2001), are IUPAC strings (IUPAC is a shorthand name for International Union of Pure and Applied Chemistry). An *IUPAC string* is simply a string over an extended DNA alphabet of size 15. A letter of an IUPAC string is called an *IUPAC nucleotide code*, or simply *IUPAC code*, and corresponds to one or more bases of the DNA alphabet. The IUPAC code is presented in Table 2.1 (page 14).

The previous representations of TFBS's make a strong assumption that binding site positions are independent of each other. To overcome such an assumption, several extensions have been proposed. First attempts extended the PSSM model to include pairs of correlated positions (O'Flanagan et al., 2005; Zhou and Liu, 2004; Benos et al., 2002). Other extensions appeared based on Markov chains. Here, a binding site is represented by a Markov chain that gives the probability of each nucleotide occurring at a particular position depending on the nucleotides at preceding positions. In this context, a n-th Markov chain was proposed to model probabilistically a motif (Lim and Burge, 2001).

A first drawback of the n-th Markov chain models is that it is hard to find a good n. A high n would give high number of parameters whereas a low n may miss out some dependencies of interest. To overcome this problem, a *variable-length Markov model* (VLMM) was proposed by Cawley (2000) to account for the variability on the relative importance of dependencies within the motif. Moreover, a second drawback of Markov chain models is that although these models allow dependencies among positions to be encoded in the state transition probabilities, not all dependencies are treated equally. Indeed, dependence between two positions is directly

IUPAC nucleotide code	**DNA base**
A	Adenine
C	Cytosine
G	Guanine
T (or U)	Thymine (or Uracil)
R	A or G
Y	C or T
S	G or C
W	A or T
K	G or T
M	A or C
B	C or G or T
D	A or G or T
H	A or C or T
V	A or C or G
N	any base

Table 2.1: The IUPAC nucleotide code with corresponding DNA bases.

represented in the Markov chain if the positions are adjacent, or within close proximity in the case of a high order Markov chain. Otherwise it is only indirectly represented. Correlation among non-adjacent positions could be especially important for TFBS's since the biding between a DNA molecule and a protein molecule is essentially a 3-dimensional geometrical matching process that may involve cooperation between nucleotides at non-adjacent positions of the primary DNA sequence. A way to circumvent this problem is to permute the positions in the binding site to maximize inter-position dependence, as measured by χ^2 values, and then define a Markov chain ordered in such a way that most pairs, or groups, in the case of high-order Markov chain, of significantly dependent positions are adjacent (Ellrott et al., 2002). Furthermore, Zhao et al. (2004) proposed that the positions in the Markov chain may also be permuted before a VLMM is applied leading to a *permuted variable-length Markov* *model* (PVLMM).

An exhaustive work exploiting inter-position dependences of TFBS's was done by Barash et al. (2003) resulting from it four new motif representations: mixtures of PSSM's, Bayesian networks, tree Bayesian networks and mixture of trees. A *mixture of PSSM's* is a simple way to enrich a PSSM by combining it with a hidden mechanism. This leads to a natural extension which takes into account that a TF can have several types of binding: slightly different physical configurations of the protein at the binding site, each with somewhat different preferences. Mixture of PSSM's capture broad dependencies among all positions via the hidden variable. On the other hand, using a *Bayesian network* it is possible to capture local dependencies by considering how each position of the binding site depends on the other. In this Bayesian network model a directed acyclic graph is used to represent such dependencies. Moreover, a *tree Bayesian network* is a sub-class of Bayesian networks where each position has at most one parent, generalizing in this way first-order Markov chains. An important advantage of tree Bayesian networks is that there exist efficient algorithms for learning the best structure (Friedman et al., 1997; Chow and Liu, 1968; Edmonds, 1967). Moreover, in contrast to first-order Markov chains, tree Bayesian networks naturally capture non-adjacent dependencies having no need to develop artificial mechanisms to capture dependencies spread out along binding site positions. Finally, tree Bayesian networks were enriched in the same way as mixtures of PSSM's leading to *mixtures of trees*. The use of mixture of trees shows to be a good compromise between the number of free parameters and the ability to model the dependencies of interest.

On the other hand, instead on focusing on dependencies between specific nucleotides at different positions, an alternative extension has been focusing on models where highly conserved positions are partially contiguous rather than evenly spread out in the motif (Xing et al., 2002). In this model there is an underlying Markov chain which favors transitions between positions with similar degrees of conservation. Another work achieves similar properties by assigning conservation types (strong, moderate and low) to blocks of motif positions (Kechris et al., 2004).

Finally, another interesting approach proposed to enrich probabilistic models with a plentiful set of features that provide superior discriminative power for TFBS's detection (Fu et al., 2009). In this method, a PSSM provided a good baseline measure for the conditional random field method employed, and some extra features were heuristically calculated, based on sequence data, or taken from external annotation. These features include architecture of the regulatory region, presence of repeats, an evolutionary score-based feature, GC-content, melting temperature, nucleosome occupancy, reverse complementarity and conservation symmetry. The advantage of this method is the fact that new features can be incorporated at will.

2.3 Motif discovery

Identifying TFBS's is notoriously difficult for both prokaryotic and eukaryotic organisms. There are two major limitations in this task. First, there is a constraint of algorithmic nature, meaning that in general the proposed methods can only be applied to sets of sequences restricted in their length and number. Second, there is a weakness in the models employed for TFBS's, leading to poor TFBS predicting methods. Nevertheless, the subject has gained a renewed interest in the last few years, with the sequencing of the genomes of vertebrates such as man and mouse. The literature on the topic of DNA binding site sequence detection is extensive, and there are several surveys on the subject (Sandve and Drablos, 2006; Tompa et al., 2005; Pavesi et al., 2004a; Stormo, 2000; Vanet et al., 1999; Brazma et al., 1998a). Herein, we concentrate on briefly surveying methods that try to extract conserved single binding sites, or multiple ones, possibly located at constrained distances from one another in a set of co-regulated DNA sequences.

Up to ten years ago, all methods for detecting DNA binding sites considered each such site individually. These methods therefore looked for *single motifs*, that is, motifs composed of a unique binding site. This includes pattern-based approaches which allowed for wildcards or a limited number of spacers but not for mutations (Brazma et al., 1998b; Tompa, 1999; van Helden et al., 1998). Apart from these, only an approach by Sagot (1998) based on a suffix tree allowed for mutations. Another very popular single motif discoverer is the MEME algorithm (Bailey and Elkan, 1994, 1995a,b), an Expectation Maximization (EM) procedure that identifies motifs with high relative entropy. MEME deals with single and *multiple motifs* by identifying significant sets of compatible motifs. It works by iteratively building such multiple motifs from single ones ensuring that the occurrence positions of the multiple motifs do not contradict the occurrence positions of the single motifs previously identified. In this case, the set of motifs reported must only satisfy compatibility. No constraint, and therefore no statistical value, is put on the distances separating them.

At that time there were few exceptions to the single motif model. An exception was a heuristic approach by Cardon and Stormo (1992) which looked for motifs composed of two parts separated by a distance which was estimated by the algorithm. Like MEME, the method is based on an EM procedure to identify sets of words with high relative entropy. Alternatively, pattern-based approaches for motif discovery were considered. In these approaches the pattern may be degenerated, that is, written on a physico-chemical alphabet, and allow for mutations (Marsan and Sagot, 2000; Vanet et al., 1999, 2000). To reflect the fact that a promoter is fragmented in several binding sites Marsan and Sagot (2000) introduced the concept of *structured motifs*. A structured motif is described as an ordered collection of *boxes*, a maximum number of substitutions allowed for each box, and an interval of distance for each pair of consecutive boxes. Meanwhile, other approaches improving the theoretical running-time of Marsan and Sagot (2000) work appeared (Carvalho, Freitas, Oliveira, and Sagot, 2006, 2005, 2004; Carvalho, 2004). Finally, there was a Fasta-inspired method which allows for spacers in general (Fraenkel et al., 1995), seeking for exact short motifs occurring in conserved order along diverse DNA sequences.

More recently, other methods have emerged which try to address the combinatorics of promoter regions. A number of algorithms have been developed that detect motifs composed of two parts, which we henceforward call *boxes*, separated by a spacer, often of fixed length (van Helden et al., 2000; Eskin and Pevzner, 2002; Eskin et al., 2003). Besides considering more complex motifs of a limited type only, with two boxes at most, the algorithms for discovering such motifs are in general naive: they either exhaustively enumerate all possible motifs of two boxes separated by a distance (van Helden et al., 2000), or discover them by crossing the lists of occurrences of single motifs detected in a previous step (Eskin and Pevzner, 2002; Eskin et al., 2003). In the first case, the method is severely limited in the length of the motifs it can identify and in the distance between them, which is usually fixed. In the second case, detecting motifs with two boxes by crossing the lists of single motifs takes time at least quadratic in the number of such single motifs and their occurrences. To address this problem, the lists for single motifs are trimmed by statistical significance before the crossing operation. However, a motif with two boxes may be statistically significant even though none of the boxes taken individually are. Indeed, one of the main interests in seeking for complex motifs directly lies in this fact.

After some years of intense research on motif discovery it become obvious that TFBS's, encoding complex regulatory signals, exhibit a high degree of degeneracy among binding sites. Sandve and Drablos (2006) pointed out that such degeneracy was the reason why previous pattern matching-based methods often suffered from impractically high false positive rates and noisy PSSM's charactering binding sites. This problem was addressed by incorporating some extra knowledge, usually carefully selected from the literature, in motif discovery methods in order to capture a variety of characteristics of the motif patterns.

Some interesting works in this line of research made use of the DNA structure for motif discovery. These works take into consideration the bendability of a region, as well as the nucleotide position in DNA loops, to determine sequence accessibility (Beiko and Charlebois, 2005; Pudimat et al., 2004; Ponomarenko et al., 1999). A quite different and particularly interesting work was extensively devised by R. Lavery et al. (Deremble and Lavery, 2005; O'Flanagan, Paillard, Lavery, and Sengupta, 2005; Paillard and Lavery, 2004; Paillard, Deremble, and Lavery, 2004; Lafontaine and Lavery, 2001a,b, 2000). In one of these works (Deremble and Lavery, 2005), the atomic structure of the protein, which specifically bounds to a fragment of DNA, was used to calculate the binding energy needed for the full combinatorial space of base sequences. Binding sites were selected considering an energy cutoff. The results suggest that the crystallographic structure of a protein-DNA complex indeed contains enough information to locate the binding sequences of a protein.

Recently, a general approach was proposed which allows for the incorporation of almost any type of information into the class of motif discovery algorithms based on Gibbs sampling (Narlikar et al., 2007). This extra information is incorporated in a *position-specific prior* (PSP) and it amounts for the likelihood that a motif starts in a certain position of a specific sequence from a set of co-regulated DNA sequences. A PSP is built in pre-processing time for that particular sequence-set and is then used to bias the optimization procedure towards real motifs. Prior information such as orthologous conservation, DNA duplex stability, nucleosome positioning and transcription factor structural class have been shown to be very effective when used with the Gibbs sampler-based PRIORITY algorithm (Narlikar et al., 2006, 2007; Gordân et al., 2008; Gordân and Hartemink, 2008; Gordân et al., 2010). Meanwhile, MEME researchers also pointed out that PSP's are beneficial when used within their EM procedure (Bailey et al., 2010).

The development of new PSP's is *per se* a hot research topic. A PSP introduces some extra knowledge taken from the literature, or computed from the sequences, that helps in discriminating real motifs from spurious ones. Whatever information it contains, computing PSP's is always a burdensome task. Indeed, the nucleosome-based PSP devised by PRIORITY researchers (Narlikar et al., 2007) assented in a discriminative view of a genome-wide organization of nucleosomes given by the work of Segal et al. (2006). The same authors also devised in a similar way another nucleosome-based PSP with the work of Lee et al. (2004). These genome-wide works have been made available for a single organism, the yeast. Therefore, a particular study, for a specific organism, usually leads to a single way of building a PSP. Having this, the PSP is built for a particular sequence-set. Notwithstanding the implied effort, nucleosome occupancy information has been shown to be of great effect on motif discovery, supporting that eukaryotic genomes are packaged into nucleosomes along chromatin affecting in this way sequence accessibility.

Additionally, the PRIORITY researchers devised another PSP prior including DNA duplex stability information (Gordân and Hartemink, 2008). This is supported by the fact that, in general, the energy needed to destabilize the DNA double helix is higher at TFBS's than at random DNA sites. But, once again, the only eukaryotic organism whose helix destabilization energy profile has been made available is yeast. Finally, orthologous conservation is the prior more broadly applied in motif discovery (Wang and Stormo, 2003; Sinha et al., 2004; Bailey and Elkan, 1995b; Harbison et al., 2009; Siddharthan et al., 2005; Kellis et al., 2003; Liu et al., 2004; MacIsaac et al., 2006; Bailey et al., 2010; Gordân et al., 2010, 2008), as information for higher organisms is now available. There are already PSP conservation-based priors for yeast, fly, mouse and even human (Bailey et al., 2010; Gordân et al., 2010, 2008). These priors have been devised by PRIORITY and MEME researchers and are based on the fact that if a particular DNA site is more conserved across related organisms then is more likely to be functional.

Meanwhile, chromatin immunoprecipitation (ChiP) followed by ultra-high-throughput sequencing, known as ChiP-seq, brought new challenges for motif discovery (Valouev et al., 2008). As a result of direct sequencing of all DNA fragments from ChiP assays, ChiP-seq is able to unravel DNA sites, across the entire genome, where a specific protein binds. Regions of high sequencing read density are referred to as *peaks* to capture the evidence of high base-specific read coverage. Peaks are found by peak finding algorithms (Fejes et al., 2008), which is called *peak calling*, yielding a set of DNA fragments of ChiP-enriched genomic regions. Usually, DNA fragments of size ±100bp are extracted around top peaks and then a motif discovery tool is used to find overrepresented sequences (Chen et al., 2008). Some authors have further exploited the information provided by these binding peaks by devising priors that use coverage profiles as motif positional preferences (Kulakovskiy et al., 2010; Hu et al., 2010).

Chapter 3

Related Work

Herein we detail related work needed to make this thesis self-contained. We introduce notation and present definitions, results and algorithms that the reader should be aware for a complete understanding of the contributions of the thesis. We advise the intended reader to have a brief tour throughout this chapter and using it only as a quick reference text.

3.1 Bayesian network models

Bayesian networks (Pearl, 1988) allow efficient and accurate representation of the joint probability distribution over a set of random variables. For this reason, they have been widely used in several domains of application where uncertainty plays an important role, like medical diagnosis and modeling DNA binding sites.

In this section we introduce some notation, while recalling relevant concepts and results concerning Bayesian networks which are directly related with the contribution of this thesis.

3.1.1 Bayesian networks

Let X be a *discrete random variable* taking values in a countable set $\mathcal{X} \subset \mathbb{R}$. In all what follows, the domain $\mathcal{X}$ is finite. We denote an n-dimensional *random vector* by $\mathbf{X} = (X_1, \ldots, X_n)$ where each component X_i is a random variable over $\mathcal{X}_i$. For each variable X_i, we denote the elements of $\mathcal{X}_i$ by $x_{i1}, \ldots, x_{ir_i}$ where r_i is the number of values X_i can take. We say that x_{ik} is the k-th value of X_i, with $k \in \{1, \ldots, r_i\}$. The probability that $\mathbf{X}$ takes value $\mathbf{x}$ is denoted by $P(\mathbf{x})$, conditional probabilities $P(\mathbf{x} \mid \mathbf{z})$ being defined correspondingly. The random vector $\mathbf{X}$ is said to be *conditionally independent* of random vector $\mathbf{Y}$ given random vector $\mathbf{Z}$ if $P(\mathbf{x} \mid \mathbf{y}, \mathbf{z}) = P(\mathbf{x} \mid \mathbf{z})$.

Definition 3.1.1 (Bayesian network) A *Bayesian network* (BN) is a triple $B = (\mathbf{X}, G, \Theta)$ where:

- $\mathbf{X} = (X_1, \ldots, X_n)$ is a random vector where each random variable X_i ranges over by a finite domain $\mathcal{X}_i$.
- $G = (\mathbf{X}, E)$ is a *directed acyclic graph* (DAG) with nodes in $\mathbf{X}$ and edges E representing direct dependencies between the variables.[1]
- $\Theta = \{\theta_{ijk}\}_{i\in\{1\ldots n\},\, j\in\{1,\ldots,q_i\},\, k\in\{1,\ldots,r_i\}}$ are the *parameters* encoding the *local distributions* of the network via

  $$P_B(X_i = x_{ik} \mid \Pi_{X_i} = w_{ij}) = \theta_{ijk},$$

  where Π_{X_i} denotes the (possibly empty) set of *parents* of X_i in G. Moreover, for each node X_i, the number of possible *parent configurations* (vectors of parent's values) is denoted by q_i. The actual parent configurations are ordered (arbitrarily) and denoted by $w_{i1}, \ldots, w_{iq_i}$ and we say that w_{ij} is the j-th configuration of Π_{X_i}, with $j \in \{1, \ldots, q_i\}$.

A Bayesian network defines a unique joint probability distribution over $\mathbf{X}$ given by

$$P_B(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_B(X_i \mid \Pi_{X_i}). \tag{3.1}$$

The conditional independence properties pertaining to the joint distribution are essentially determined by the network structure. Specifically, X_i is conditionally independent of its non-descendants given its parents Π_{X_i} in G (Pearl, 1988).

For convenience, we introduce a few additional notations that apply to Bayesian network models intended to be learned from data T. N_{ijk} is the number of instances in the data T where the variable X_i takes its k-th value x_{ik} and the variables in Π_{X_i} take their j-th configuration w_{ij}. N_{ij} is the number of instances in the data T where the variables in Π_{X_i} take their j-th configuration w_{ij}, that is,

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk}.$$

[1]For the sake of simplicity we do not distinguish the random vector $\mathbf{X} = (X_1, \ldots, X_n)$ from the set of random variables $\{X_1, \ldots, X_n\}$.

Moreover, the total number of instances in the data T is N. Finally, we denote the set of all Bayesian networks with n variables by $\mathcal{B}_n$.

The problem of learning a Bayesian network given data T consists in finding the Bayesian network that best fits the data T. There are three approaches for learning Bayesian networks (see Koller and Friedman, 2009), namely, *constraint-based learning*, *score-based learning* and *Bayesian model averaging*. In this thesis we are particularly interested in score-based learning, where a *scoring criterion* ϕ is considered in order to quantify the fitting of a Bayesian network. In this context, the problem of learning a Bayesian network can be paraphrased in the following optimization problem.

Definition 3.1.2 (Learning a Bayesian network) Given a data $T = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ and a scoring criterion ϕ, the *problem of learning a Bayesian network* is to find a Bayesian network $B \in \mathcal{B}_n$ that maximizes the value $\phi(B,T)$.

Contributions in this area of research are typically divided in two different problems: *scoring* and *searching*. The scoring problem focus on devising new scoring criteria to measure the goodness of a certain network structure given the data. On the other hand, the searching problem concentrates on identifying one or more network structures that yield a high value for the scoring criterion in mind. If the search is conducted with respect to a neighborhood structure defined on the space of possible solutions then we are in the presence of *local score-based learning*. Local score-based learning algorithms can be extremely efficient if the scoring criterion employed is decomposable.

Definition 3.1.3 (Decomposable scoring criterion) A scoring criterion ϕ is *decomposable* if the score assigned to each network decomposes over the network structure in such a way that it can be expressed as a sum of local scores ϕ_i that depends only on each node X_i and its parents, that is, scores of the following form:

$$\phi(B,T) = \sum_{i=1}^{n} \phi_i(\Pi_{X_i}, T).$$

Well known decomposable scores are those based on information theory, such as *log-likelihood* (LL), *Akaike information criterion* (AIC) (Akaike, 1974), *Bayesian information criterion* (BIC), also called as *minimum description length* (MDL) (Lam and Bacchus, 1994; Suzuki, 1993), *factorized conditional log-likelihood* (fNML) (Kontkanen and Myllymäki, 2007;

Roos et al., 2008) and Bayesian scoring function such as K2 (Cooper and Herskovits, 1992), *Bayesian Dirichlet* (BD) and its variants (BDe and BDeu) (Heckerman et al., 1995; Buntine, 1991) and *mutual information tests* (MIT) (de Campos, 2006).

Learning unrestricted Bayesian networks from data under typical scoring criteria is NP-hard (Chickering et al., 2004). This result led the community to search for the largest subclass of Bayesian networks for which there is an efficient learning algorithm. First attempts confined the network to tree structures and used Edmonds (1967) and Chow and Liu (1968) optimal branching algorithms to learn the network. More general classes of Bayesian networks have eluded efforts to develop efficient learning algorithms. Indeed, Chickering (1996) showed that learning the structure of a Bayesian network is NP-hard even for networks constrained to have in-degree at most 2. Later, Dasgupta (1999) showed that even learning an optimal *polytree* – that is, a DAG in which there are not two different paths from one node to another – where each node has at most in-degree 2 is NP-hard. Moreover, Meek (2001) showed that identifying the best *path structure* – that is, a total order over the nodes – is hard. Due to these hardness results exact polynomial-time bounded approaches for learning Bayesian networks have been restricted to tree structures.

Consequently, the standard methodology for addressing the problem of learning Bayesian networks became heuristic search, based on scoring metrics optimization, conducted over some search space. Many search algorithms have been proposed along these lines, varying both on the formulation of the search space (network structures, equivalence classes of network structures and orderings over the network variables), and on the algorithm to search the space (greedy hill-climbing, simulated annealing, genetic algorithms, tabu search, etc). The most common scoring criteria employed in Bayesian-network learning are reviewed in Carvalho (2009) and Yang and Chang (2002). We refer the interested reader to newly developed scoring criteria to the works of de Campos (2006) and Silander, Roos, Kontkanen, and Myllymäki (2008). A review concerning Bayesian scores can also be found in Heckerman, Geiger, and Chickering (1995).

Scoring criteria for learning Bayesian networks

Herein, we present the foundations of all the scoring criteria needed in this thesis from an information-theoretic point of view. In this study we include scoring functions based on

Rissanen's stochastic complexity (Rissanen, 1986, 1987, 1989, 1995a,b, 1996) that were developed recently (Kontkanen and Myllymäki, 2007; Roos, Silander, Kontkanen, and Myllymäki, 2008; Silander, Roos, Kontkanen, and Myllymäki, 2008). We refer the reader interested in Rissanen's stochastic complexity to the review of Lanterman (2001).

Information-theoretic scoring functions are based on compression. In this context, the score of a Bayesian network B is related to the compression that can be achieved over the data T with an optimal code induced by B. The overall idea is to choose a representation of the data which permits to express it with the shortest possible length. In theory, this can be achieved with Kolmogorov complexity, that is, with the shortest computer program which generates the data T. Unfortunately, this result is of little practical use since Kolmogorov complexity is incomputable, that is, there is no algorithm which can find the shortest computer program to generate a particular data, or even find the length of such a shortest program. To avoid such an entanglement one may consider minimizing the description length over a set of candidate hypothesis H (since finding the length of the absolutely shortest possible program would be futile). We do not need to suppose that the data are the result of a realization of one of the models. Indeed, as more models are developed they can be added to the set of hypothesis.

Given data T and a set of probability distributions $\mathcal{H}$ that may be used to describe T, we take the *length of describing* T *with* H to be the sum

$$L(T, H) = L(T \mid H) + L(H),$$

where $L(T \mid H)$ is the length (in bits) of the description of T when encoded with H and $L(H)$ is the length of the description of H. Shannon's theory tell us that for a given hypothesis H, we can construct a code for T with length $L(T \mid H) = -\mathrm{LL}(H \mid T) = -\log P(T \mid H)$ where $P(T \mid H)$ is the probability of sampling T with distribution H. Fortunately, it is not necessary to construct such codes. We only need to know that it is possible and to have expressions for its length. If the hypothesis H could somehow be transmitted cost-free then it is enough to choose the hypothesis H that minimizes $L(T \mid H)$, that is, the *maximum likelihood* (ML) estimate. In this case, $L(T, H) = L(T \mid H) = -\mathrm{LL}(H \mid T)$ and we obtain the *log-likelihood* scoring criterion. On the other hand, the *minimum description length principle* imposes that the parameters of H need also to be transmitted. In general, different sets of choices for $L(H)$ will yield different expressions. Next, we describe the two main contributions to compute $L(H)$ when H is a Bayesian network B.

The first approach assumes that only integers are used to encode the parameters of B. In this case the optimal *(universal) code for integer* numbers (Rissanen, 1983, 1987) is such that the length of an integer x would take

$$\log^*(x) + \log(c)$$

bits, with $c \approx 2.865064$, and where $\log^*$ is defined as

$$\log^*(x) = \log(x) + \log\log(x) + \log\log\log(x) + \dots$$

If x is a nonnegative real number, like the parameters of a Bayesian network B, then the real x should be represented by an integer $\frac{x}{\delta_x}$ where δ_x is the precision of the representation. It is shown that by approximating $\log^* \approx \log$ it is possible to compute the optimal δ_x, say d. Moreover, by taking the number of independent samples $N \to \infty$ we have that the length of a real x would take

$$\log^*\left(\frac{x}{d}\right) + \log(c) \to \frac{1}{2}\ln(N),$$

where $\log(.)$ is the binary logarithm and $\ln(.)$ is the natural logarithm. Thus, the number of bits required to represent a Bayesian network B is

$$\frac{1}{2}\ln(N)|B|$$

where $|B|$ is the total number of parameters of B. Observe that $|B|$ decomposes over the network structure since the parameters associated to a node X_i are θ_{ijk} and they sum up to $(r_i - 1) \times q_i$. Indeed, for a node X_i there are q_i multinomials ranging over by r_i values, and a multinomial over r values has only $r - 1$ degrees of freedom. This approach led to the development of the *minimum description length* (MDL) scoring criterion defined as follows

$$\mathrm{MDL}(B \mid T) = -\mathrm{LL}(B \mid T) + \frac{1}{2}\ln(N)|B|.$$

This scoring criterion coincides with the *Bayesian information criterion* (BIC), also known as *Schwarz Bayesian Criterion* (SBC), developed by Schwarz (1978) from a Bayesian perspective point of view.

Alternatively, to compute $L(H)$ by representing parameters via a universal encoding of the integers, Barron, Rissanen, and Yu (1998) proposed an approach based on Rissanen's *stochastic complexity* (Rissanen, 1986, 1987, 1989, 1995a,b, 1996), which does not explicitly encode the parameters. If data T of size N was encoded with an hypothesis H, then the *regret of* H, relative to a set of hypothesis $\mathcal{H}$ over all data of fixed size N, is the number of extra bits required to encode T using an optimal code in $\mathcal{H}$. The idea is to find $H \in \mathcal{H}$ that minimizes the worst-case regret over all data of fixed size N. Shtarkov (1997) showed how to compute the solution to this *minmax problem*, and the resulting distribution was called *normalized maximum likelihood* (NML). The length of the associated code is given by $-\mathrm{LL}(H \mid T) + \mathcal{C}_T(H)$ where $\mathcal{C}_T(H)$ is called the *parametric complexity of* H *for data* T. Although it is hard to compute $\mathcal{C}_T(H)$ in general, there are tractable formulas for a handful of models (Grünwald, 2007). In the context of data of size N_{ij}, generated by a multinomial ranging over by r_i values, as for the case of attribute X_i given its parents Π_{X_i} in a Bayesian network, a recursive formula was found by Kontkanen and Myllymäki (2007). This result allows to construct a decomposable penalization for Bayesian networks since each node accounts for q_i multinomials ranging over by r_i values. Thus, the local penalty associated to the node X_i is given by

$$\sum_{j=1}^{q_i} \mathcal{C}^{r_i}_{N_{ij}}$$

where $\mathcal{C}^{r_i}_{N_{ij}}$ is the parametric complexity associated to data of size N_{ij} generated by a multinomial ranging over by r_i values. This approach motivated the *factorized normalized maximum likelihood* (fNML) scoring criterion (see Kontkanen and Myllymäki, 2007; Roos, Silander, Kontkanen, and Myllymäki, 2008; Silander, Roos, Kontkanen, and Myllymäki, 2008) whose expression is as follows

$$\mathrm{fNML}(B \mid T) = -\mathrm{LL}(B \mid T) + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \mathcal{C}^{r_i}_{N_{ij}}.$$

We now give all details to compute $C^{r_i}_{N_{ij}}$ for a general Bayesian network. By Kontkanen and Myllymäki (2007), the parametric complexity $\mathcal{C}^{r}_{m}$ can be computed as follows. Let

$$M = \{N_{ij} : 1 \leq i \leq n, 1 \leq j \leq q_i\}$$

and $R = \max_{i=1,\dots,n} r_i$. For reasonable R and $|M|$, where $|M|$ is the cardinality of the set M, these values can be stored in a $|M| \times R$ table, called a $\mathcal{C}$*-table*, which can be computed once as a pre-processing step before structure learning. The computation of the $\mathcal{C}$-table is as follows. For all $m \in M$, $\mathcal{C}_m^1 = 1$. For all $1 \leq r \leq R$, $\mathcal{C}_0^r = 1$. Moreover, for $r = 2$,

$$\mathcal{C}_m^2 = \sum_{h=0}^{m} \binom{m}{h} \left(\frac{h}{m}\right)^h \left(\frac{m-h}{m}\right)^{m-h},$$

and, for $2 < r \leq R$,

$$\mathcal{C}_m^r = \mathcal{C}_m^{r-1} + \frac{m}{r-2}\, \mathcal{C}_m^{r-2}. \tag{3.2}$$

For very large $|M|$, computing columns $\mathcal{C}_m^2$, for all $m \in M$, may be prohibitive. For that case, a very accurate Szpankowski approximation

$$\mathcal{C}_m^2 = \frac{m\pi}{2} e^{\sqrt{\frac{8}{9m\pi}} + \frac{3\pi-16}{36m\pi}}$$

can be used (Kontkanen, Buntine, Myllymäki, Rissanen, and Tirri, 2003) making the computation more efficient. If the space for storing the $\mathcal{C}$-table is critical, Silander, Roos, Kontkanen, and Myllymäki (2008) propose to store only the 1000 first entries of the column C_m^2, for all $m \in M$, use Szpankowski approximation for the rest of the column, and use formula (3.2) for computing the values for $2 < r \leq R$.

Score equivalent scoring functions

Two Bayesian-network structures are said to be *equivalent* if the set of distributions that they can represented is precisely the same. Many scoring criteria that are used to learn Bayesian-network structures from data are *score equivalent*, that is, these scoring criteria assign the same score to equivalent structures. Chickering (2002) argued that score equivalence is a desirable property of a scoring criterion, unless extra semantics, such as causability, can be attributed to the edges of a Bayesian network. Other authors (Yang and Chang, 2002; de Campos, 2006), however, concluded that, in practice, non-score-equivalent scoring criteria perform better than score-equivalent ones.

The formal definition of score-equivalent scoring criteria needs some background in graph theory which is introduced next.

Two variables X and Y are *adjacent* if there is an edge between X and Y.

Definition 3.1.4 (v-structure) In a directed acyclic graph, a *v-structure* is a local dependency $X \rightarrow Z \leftarrow Y$ such that X and Y are not adjacent.

Theorem 3.1.5 (Verma and Pearl (1990)) Two directed acyclic graphs are equivalent if and only if they have the same skeleton and the same v-structures.

By Theorem 3.1.5, all tree-network structures with the same skeleton are equivalent, regardless of the direction of the edges.

Because DAG equivalence is reflexive, symmetric, and transitive, it defines a set of equivalence classes over DAG's. One way to represent the equivalence class of equivalent DAG's is by the means of a partially directed acyclic graph.

Definition 3.1.6 (Partially directed acyclic graph) A *partially directed acyclic graph* (PDAG) is a graph which contains both directed and undirected edges, with no directed cycle in its directed subgraph.

From Theorem 3.1.5, it follows that a PDAG containing a directed edge for every edge participating in a v-structure, and an undirected edge for every other edge, uniquely identifies an equivalence class of DAG's. There may be many other PDAG's, however, that correspond to the same equivalence class. For example, any DAG interpreted as a PDAG can be used to represent its own equivalence class.

Definition 3.1.7 (Compelled edge) A directed edge $X \to Y$ is *compelled* in a directed acyclic graph G if for every directed acyclic graph G' equivalent to G, $X \to Y$ exists in G'.

By Theorem 3.1.5, all edges participating in a v-structure are compelled. Not every compelled edge, however, necessarily participates in a v-structure. For example, the edge $Z \to W$ in the DAG with edges $E = \{X \to Z, Z \to W, Y \to Z, Y \to U\}$ is compelled. Moreover, for any edge e in G, if e is not compelled in G, then e is *reversible* in G. In that case, there exists some DAG G' equivalent to G in which e has opposite direction.

Definition 3.1.8 (Essential graph) An *essential graph*, denoting an equivalence class of directed acyclic graphs, is the partially directed acyclic graph consisting of a directed edge for every compelled edge in the equivalence class, and an undirected edge for every reversible edge in the equivalence class.

Essential graphs are used to represent equivalent class of network structures during Bayesian network learning. The essential graph of a tree-network structure is its skeleton.

Definition 3.1.9 (Score-equivalent scoring function) A scoring function ϕ is *score equivalent* if it assigns the same score to all directed acyclic graphs that are represented by the same essential graph.

All interesting scoring criteria in the literature are decomposable, since it is unfeasible to learn undecomposable scores. LL and MDL scoring criteria are decomposable and score equivalent (Hastie et al., 2003), whereas the fNML scoring criterion is decomposable but not score equivalent (Silander et al., 2008).

Chow-Liu tree learning algorithm

A *tree Bayesian network* is a Bayesian network where the underlying DAG is a directed tree. Finding the tree Bayesian network that maximizes the LL score given data T can be done in polynomial time by the Chow and Liu (1968) tree learning algorithm.

In order to understand how to solve the learning problem for tree Bayesian networks we need to formulate the $\mathrm{LL}(B \mid T)$ using mutual information (Bouckaert, 1995). Start by considering that $T = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, where the t-th instance of T is given by $\mathbf{y}_t = (y_t^1, \ldots, y_t^n)$. Applying Equation (3.1), page 22, to the log-likelihood given by

$$\mathrm{LL}(B \mid T) = \sum_{t=1}^{N} \log(P_B(y_t^1, \ldots, y_t^n))$$

we obtain (see Heckerman et al., 1995) that

$$\mathrm{LL}(B \mid T) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log(\theta_{ijk}). \tag{3.3}$$

When the structure of the network is fixed in advance maximizing the likelihood of the data T reduces to estimating the parameters θ_{ijk}. In this case, the *maximum likelihood* (ML) parameters that maximize LL are simply the *observed frequency estimates* (OFE) given by

$$\hat{\theta}_{ijk} = \hat{P}_T(X_i = x_{ik} \mid \Pi_{X_i} = w_{ij}) = \frac{N_{ijk}}{N_{ij}}, \tag{3.4}$$

Therefore, plugging these estimates back in the LL scoring criterion yields

$$\widehat{\mathrm{LL}}(G \mid T) \quad = \quad \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log\left(\frac{N_{ijk}}{N_{ij}}\right). \tag{3.5}$$

The notation with G as the argument instead of $B = (\mathbf{X}, G, \Theta)$ emphasizes that once the OFE parameters are decided upon, the criterion is a function of the network structure, G, only.

Assuming that the parameters that maximize the LL fulfill Equation (3.4), the LL in Equation (3.3) can be rewritten as follows:

$$\begin{aligned}\widehat{\mathrm{LL}}(G \mid T) &= -N \sum_{i=1}^{n} H_{\hat{P}_T}(X_i \mid \Pi_{X_i}) \\ &= N \sum_{i=1}^{n} I_{\hat{P}_T}(X_i; \Pi_{X_i}) - N \sum_{i=1}^{n} H_{\hat{P}_T}(X_i), \qquad (3.6)\end{aligned}$$

where $I_{\hat{P}_T}$ is the *mutual information*[2] and $H_{\hat{P}_T}$ is the *entropy*[3] (for more details about these quantities see Cover and Thomas, 2006). The subscript $\hat{P}_T$ indicates that the information theoretic quantities are evaluated under the joint distribution $\hat{P}_T$ of $\mathbf{X}$ induced by the OFE parameters. Observe that the right-hand side of (3.6) has two terms and only the first depends on the network structure G, hence, maximizing $\widehat{\mathrm{LL}}(G \mid T)$ resumes to maximizing

$$\sum_{i=1}^{n} I_{\hat{P}_T}(X_i; \Pi_{X_i}) = \sum_{\substack{i=1 \\ i \neq R}}^{n} I_{\hat{P}_T}(X_i; X_{\pi(i)})$$

where R is the root of the tree Bayesian network B and $\pi(i)$ is the index of the parent variable of X_i, that is, $\Pi_{X_i} = \{X_{\pi(i)}\}$ for $i \neq R$. Recall that the *mutual information of two random vectors* is given by

$$I(\mathbf{X}; \mathbf{Y}) = \sum_{\mathbf{x},\mathbf{y}} P(\mathbf{x}, \mathbf{y}) \log \frac{P(\mathbf{x}, \mathbf{y})}{P(\mathbf{x})P(\mathbf{y})}. \qquad (3.7)$$

The main idea of the algorithm to learn tree Bayesian networks is to consider a complete weighted undirected graph, where each undirected edge between X_i and X_j is weighted with the mutual information between X_i and X_j. Given this, the problem reduces to determining a maximal weighted (undirected) *spanning tree* – a tree composed of all nodes and some of the edges of the original (undirected) graph. After computing such spanning tree, a direction has to be assigned to each edge of the tree. This is done by choosing an arbitrary node as the tree root and then setting the direction of all edges to be outward from it. The detail of the algorithm is depicted in Algorithm 3.1.

[2]The *mutual information* between X_i and Π_{X_i}, denoted by $I(X_i; \Pi_{X_i})$, measures the mutual dependence between X_i and Π_{X_i}.

[3]The entropy of X_i, denoted by $H(X_i)$, amounts for the expected value of its *self-information*, that is, $H(X_i) = I(X_i; X_i)$. The *conditional entropy* of X_i given its parents Π_{X_i}, denoted by $H(X_i|\Pi_{X_i})$, measures the entropy of X_i when the value of the parent configuration Π_{X_i} is known.

Algorithm 3.1 Chow-Liu tree learning algorithm, LL score

1. Compute the mutual information $I_{\hat{P}_T}(X_i; X_j)$ between each pair of attributes, with $i \neq j$ and $i, j \leq n$, given by Equation (3.7).
2. Build a complete undirected graph with attributes $X_1, \ldots, X_n$ as nodes. Annotate the weight of the edge connecting X_i to X_j by $I_{\hat{P}_T}(X_i; X_j)$.
3. Build a maximal weighted (undirected) spanning tree.
4. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it and return the resulting tree.

The resulting directed tree is called *Chow-Liu tree* or *optimal branching*. Chow and Liu (1968) showed that Algorithm 3.1 is linear on the size of the data T and quadratic on the number of variables of the Bayesian network.

Theorem 3.1.10 (Chow and Liu (1968)) Let T be a collection of N instances of $X_1, \ldots, X_n$. Algorithm 3.1 constructs an optimal branching B that maximizes $\mathrm{LL}(B \mid T)$ in $O(n^2 N)$ time.

Extending Chow-Liu tree learning algorithm

The Chow-Liu tree learning algorithm was originally proposed for maximizing the LL score but it can be easily adapted to deal with any scoring function that is decomposable and/or score equivalent.

According to Heckerman et al. (1995), finding an optimal branching for decomposable and score equivalent scoring functions reduces to weighting each undirected edge between X_i and X_j by $\phi_j(\{X_i\}, T) - \phi_j(\emptyset, T)$, which is equal to $\phi_i(\{X_j\}, T) - \phi_i(\emptyset, T)$ by score equivalence of ϕ, and to find a maximal weighted (undirected) spanning tree. The detailed algorithm for learning tree Bayesian networks for decomposable and score-equivalent scoring functions is presented in Algorithm 3.2.

Algorithm 3.2 Learning tree BN's, decomposable and score equivalent ϕ–score

1. Compute $\phi_j(\{X_i\}, T) - \phi_j(\emptyset, T)$ between each pair of attributes X_i and X_j, with $i \neq j$ and $i, j \leq n$.
2. Build a complete undirected graph with attributes $X_1, \ldots, X_n$ as nodes. Annotate the weight of the edge connecting X_i to X_j by the value computed in the previous step.
3. Build a maximal weighted (undirected) spanning tree.
4. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it and return the resulting tree.

Learning an optimal branching for scoring functions that are only decomposable, but not score equivalent, can also be done in polynomial time (Heckerman et al., 1995). In this case, however, an edge between X_i and X_j may score differently depending on its direction, and so a directed spanning tree must be found (instead of an undirected one). The idea is to weight each directed edge from X_i to X_j with $\phi_j(\{X_i\}, T) - \phi_j(\emptyset, T)$ and then find an optimal directed spanning tree with Edmonds' algorithm (Edmonds, 1967; Lawler, 1976). The detailed algorithm for learning tree Bayesian networks for scoring functions that are only decomposable, but not score-equivalent, is presented in Algorithm 3.3.

Algorithm 3.3 Learning tree BN's, decomposable and non-score equivalent ϕ–score

```
1. Compute $\phi_j(\{X_i\}, T) - \phi_j(\emptyset, T)$ for each edge from $X_i$ to $X_j$, with $i \neq j$ and $i, j \leq n$.
2. Build a complete directed graph with attributes $X_1, \ldots, X_n$ as nodes. Annotate the weight of the edge from $X_i$ to
   $X_j$ by the value computed in the previous step.
3. Build a maximal weighted directed spanning tree and return it.
```

3.1.2 Bayesian network classifiers

Bayesian networks have been widely used in the context of classification (Su and Zhang, 2006; Grossman and Domingos, 2004; Friedman et al., 1997; Duda and Hart, 1973). Herein, we introduce the concept of Bayesian network classifier and then present two classifiers particularly interesting in the context of this thesis: *augmented Naive Bayes* and *Tree Augmented Naive Bayes* classifiers.

Definition 3.1.11 (Bayesian network classifier) A *Bayesian network classifier* (BNC) is a Bayesian network where $\mathbf{X} = (X_1, \ldots, X_n, C)$. The variables $X_1, \ldots, X_n$ are called *attributes*, or *features*, and C is called the *class variable.*

The Naive Bayes (NB) classifier (Duda and Hart, 1973) is one of the simplest BNC's.

Definition 3.1.12 (Naive Bayes classifier) A *naive Bayes* (NB) classifier is a Bayesian network classifier where each attribute has the class variable as its unique parent, that is, $\Pi_{X_i} = \{C\}$ for all $1 \leq i \leq n$.

The NB classifier has shown to be very effective, in the sense that, in many cases, its predictive performance is competitive with state-of-the-art classifiers (Domingos and Pazzani, 1996a,b). In addition, the NB classifier is computationally undemanding as it has a fixed graph structure, and so learning the network reduces to computing the OFE. However, the independence assumption is too strict and relaxing this assumption may lead to more accurate classifiers.

Augmented naive Bayes classifier

For efficiency purposes it is common to restrict the dependencies between the attributes and the class variable, imposing all attributes to have the class variable as parent.

Definition 3.1.13 (Augmented Naive Bayes classifier) An *augmented naive Bayes classifier* is a Bayesian network classifier where the graph structure G is such that the class variable has no parents, that is, $\Pi_C = \emptyset$, and all attributes have at least the class variable as parent, that is, $C \in \Pi_{X_i}$.

Loosely speaking, the problem of learning an augmented naive Bayes classifier can be paraphrased as the problem of learning a Bayesian network where all attributes have the class variable as parent. In this thesis we focus our attention only on augmented naive Bayes network classifiers, referring abusively to them as Bayesian network classifiers.

Tree augmented naive Bayes classifier

The tree augmented naive (TAN) Bayes classifier was proposed by Friedman, Geiger, and Goldszmidt (1997) to overcome the strong independence assumptions imposed by the NB network. In fact, the TAN is an extension of NB which allows additional edges between the attributes of the network in order to capture correlations among them. Such correlations are however restricted to a tree structure.

Definition 3.1.14 (Tree augmented naive Bayes classifier) A *tree augmented naive Bayes classifier* (TAN) is a Bayesian network classifier where there exists a root $R \in \{1, \ldots, n\}$ such that $\Pi_{X_R} = \{C\}$ and $\Pi_{X_i} = \{C, X_j\}$ for all $1 \leq i \leq n$ with $i \neq R$.

For convenience, we introduce a few additional notations that apply to augmented naive Bayes models. Recall that the parents of X_i are denoted by Π_{X_i}. The parents of X_i without the class variable are denoted by $\Pi^*_{X_i} = \Pi_{X_i} \setminus \{C\}$.

In order to understand how to solve the learning problem for TAN classifiers we need to reformulate the $\widehat{\mathrm{LL}}(G \mid T)$ using mutual information as in Equation (3.6) at page 31. With TAN models, however, we have to consider the class variable and so,

$$\begin{aligned} \widehat{\mathrm{LL}}(G \mid T) &= -N \sum_{i=1}^{n} H_{\hat{P}_T}(X_i \mid \Pi^*_{X_i}, C) \\ &= N \sum_{i=1}^{n} I_{\hat{P}_T}(X_i; \Pi^*_{X_i}, C) - N(H_{\hat{P}_T}(C) + \sum_{i=1}^{n} H_{\hat{P}_T}(X_i)) \end{aligned} \tag{3.8}$$

Observe that the right-hand side of (3.8) has two terms and only the first depends on the network structure G, hence, maximizing $\widehat{\mathrm{LL}}(G \mid T)$ resumes to maximizing

$$\sum_{i=1}^{n} I_{\hat{P}_T}(X_i; \Pi^*_{X_i}, C). \tag{3.9}$$

We can simplify (3.9) using the chain law for mutual information,

$$I(\mathbf{X}; \mathbf{Y}, \mathbf{Z}) = I(\mathbf{X}; \mathbf{Z}) + I(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}),$$

and derive

$$\sum_{i=1}^{n} I_{\hat{P}_T}(X_i; C) + \sum_{\substack{i=1 \\ i \neq R}}^{n} I_{\hat{P}_T}(X_i; \Pi^*_{X_i} \mid C). \tag{3.10}$$

Finally, note that the first term of (3.10) does not depend on the choice of the parents $\Pi^*_{X_i}$, therefore, maximizing $\mathrm{LL}(B \mid T)$ is equivalent to maximize

$$\sum_{\substack{i=1 \\ i \neq R}}^{n} I_{\hat{P}_T}(X_i; \Pi^*_{X_i} \mid C). \tag{3.11}$$

Recall that the *conditional mutual information* is given by

$$I(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}) = \sum_{\mathbf{x},\mathbf{y},\mathbf{z}} P(\mathbf{x}, \mathbf{y}, \mathbf{z}) \log \frac{P(\mathbf{x}, \mathbf{y} \mid \mathbf{z})}{P(\mathbf{x} \mid \mathbf{z}) P(\mathbf{y} \mid \mathbf{z})}. \tag{3.12}$$

It is now easy to find the TAN that maximizes the LL score for some data T. The main idea is to consider a complete weighted undirected graph, where each edge between X_i and X_j is weighted with the conditional mutual information between X_i and X_j given the class variable C. Given this, the problem reduces to determining a maximal weighted (undirected) spanning tree. After computing such spanning tree, a direction has to be assigned to each edge of the tree. This is done by choosing an arbitrary attribute as the tree root and then

Algorithm 3.4 Learning TAN BNC's, LL score

```
1. Compute $I_{\hat{P}_T}(X_i; X_j \mid C)$ between each pair of attributes, with $i \neq j$ and $i, j \leq n$, given by Equation (3.12).
2. Build a complete undirected graph with attributes $X_1, \ldots, X_n$ as nodes. Annotate the weight of the edge connecting
   $X_i$ to $X_j$ by $I_{\hat{P}_T}(X_i; X_j \mid C)$.
3. Build a maximal weighted (undirected) spanning tree.
4. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all
   edges to be outward from it.
5. Construct a TAN classifier by adding a node labeled by $C$ and adding an arc from $C$ to each $X_i$, $i \leq n$.
```

setting the direction of all edges to be outward from it. The TAN classifier is then built by adding a node labeled by C, and adding an arc from C to each tree node. The detail of the algorithm is depicted in Algorithm 3.4.

The proof of soundness of Algorithm 3.4 follows from the derivation that led to Equation (3.11) and from the fact that we are computing a maximal weighted spanning tree. Since the step that consumes asymptotically more time is weighting the edges, Algorithm 3.4 is linear on the size of the data T and quadratic on the number of variables of the Bayesian network.

Theorem 3.1.15 (Friedman, Geiger, and Goldszmidt (1997)) Let T be a collection of N instances of $X_1, \ldots, X_n$. Algorithm 3.4 constructs a TAN classifier B that maximizes $\mathrm{LL}(B \mid T)$ in $\mathrm{O}(n^2 N)$ time.

Extending tree augmented naive Bayes classifier

The TAN learning algorithm was originally proposed for maximizing the LL score but it can be easily adapted to deal with any scoring function that is decomposable and score equivalent.

Finding an optimal TAN classifier for decomposable and score equivalent scoring functions reduces to weighting each undirected edge between X_i and X_j by $\phi_j(\{X_i, C\}, T) - \phi_j(\{C\}, T)$, which is equal to $\phi_i(\{X_j, C\}, T) - \phi_i(\{C\}, T)$ by score equivalence of ϕ, and to find a maximal weighted (undirected) spanning tree (Heckerman et al., 1995). The detailed algorithm for learning TAN classifiers for decomposable and score-equivalent scoring functions is presented in Algorithm 3.5.

Learning an optimal TAN classifier for scoring functions that are only decomposable, but not score equivalent, can also be done in polynomial time (Heckerman et al., 1995). In this

Algorithm 3.5 Learning TAN BNC's, decomposable and score equivalent ϕ–score

```
1. Compute $\phi_j(\{X_i, C\}, T) - \phi_j(\{C\}, T)$ between each pair of attributes $X_i$ and $X_j$, with $i \neq j$ and $i, j \leq n$.
2. Build a complete undirected graph with attributes $X_1, \ldots, X_n$ as nodes. Annotate the weight of the edge connecting $X_i$ to $X_j$ by the value computed in the previous step.
3. Build a maximal weighted (undirected) spanning tree.
4. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it.
5. Construct a TAN classifier by adding a node labeled by $C$ and adding an arc from $C$ to each $X_i$, $i \leq n$.
```

case, however, an edge between X_i and X_j may score differently depending on its direction, and so a directed spanning tree must be found (instead of an undirected one). The idea is to weight each directed edge from X_i to X_j with $\phi_j(\{X_i, C\}, T) - \phi_j(\{C\}, T)$ and then find an optimal directed spanning tree with Edmonds' algorithm (Edmonds, 1967; Lawler, 1976). The detailed algorithm for learning TAN classifiers for scoring functions that are only decomposable, but non-score equivalent, is presented in Algorithm 3.6.

Algorithm 3.6 Learning TAN BNC's, decomposable and non-score equivalent ϕ–score

```
1. Compute $\phi_j(\{X_i, C\}, T) - \phi_j(\{C\}, T)$ for each edge from $X_i$ to $X_j$, with $i \neq j$ and $i, j \leq n$.
2. Build a complete directed graph with attributes $X_1, \ldots, X_n$ as nodes. Annotate the weight of the edge from $X_i$ to $X_j$ by the value computed in the previous step.
3. Build a maximal weighted directed spanning tree.
4. Construct a TAN classifier by adding a node labeled by $C$ and adding an arc from $C$ to each $X_i$, $i \leq n$.
```

3.2 Extraction of structured motifs

In this section, we provide some background on the approaches used to solve the structured motif extraction problem. First, we present two relevant data structures: (i) *suffix trees*, which are widely used in string processing problems; (ii) *factor trees*, which appeared as a more compact version of suffix trees, that index only the sufficient data required by problems such as motif discovery. Next, we outline an algorithm devised by Sagot (1998) for the extraction of single motifs (SPELLER). Finally, we present an algorithm proposed by Carvalho, Freitas, Oliveira, and Sagot (2006) for structured motif extraction (RISO).

3.2.1 Basic data structures

Suffix tree

A suffix tree is a data structure built over all suffixes of a string. Such a data structure exposes the internal structure of a string and is often used to solve many string problems. The construction of a suffix tree in linear-time is a problem already addressed by Weiner (1973), by McCreight (1976), and more recently by Ukkonen (1995).

We define a suffix tree for an arbitrary string S of length n over an alphabet Σ as presented by Gusfield (1997). After that we generalize the suffix tree to handle a set of strings.

Definition 3.2.1 (Suffix tree) A *suffix tree* $\mathcal{T}$ of a n-character string S is a rooted directed tree with exactly n leaves, numbered 1 to n. Each internal node, other than the root R, has at least two children and each edge is labeled with a nonempty substring of S. No two edges out of a node can have edge-labels beginning with the same character. The key feature of the suffix tree is that for any leaf i, the label of the path from the root to the leaf i spells out exactly the suffix of S that starts at position i.

The previous definition of a suffix tree does not guarantee the existence of a suffix tree for any string S. The problem is that if a prefix of a suffix of S matches a suffix of S, the path for the later suffix would not end at a leaf. To avoid this problem we place at the end of S a special symbol that is not in the alphabet. Herein, we use the symbol \$ for the termination character. As an example, the suffix tree for string S =AGACAGGAGGC\$, over the DNA alphabet Σ={A,C,G,T}, is presented in Figure 3.1.

The suffix tree construction for a set of strings, called a *generalized suffix tree*, can be easily achieved by consecutively building the suffix tree for each string of the set. The resulting suffix tree is built in time proportional to the sum of all the string lengths. A way to distinguish the different strings in the generalized suffix tree is to convert the leaf number of the single string suffix tree to two numbers, one identifying the string and the other identifying the starting position in that string. As an example, the generalized suffix tree for strings S_1=TACTA\$ and S_2=CACTCA\$, over the DNA alphabet Σ={A,C,G,T}, is presented in Figure 3.2.

Contrarily to the motif localization problem, in the motif extraction problem the starting position of a suffix in a string is not relevant, only the identification of the string in the input set is required. A usual way to distinguish the input strings, in a motif extraction problem, is

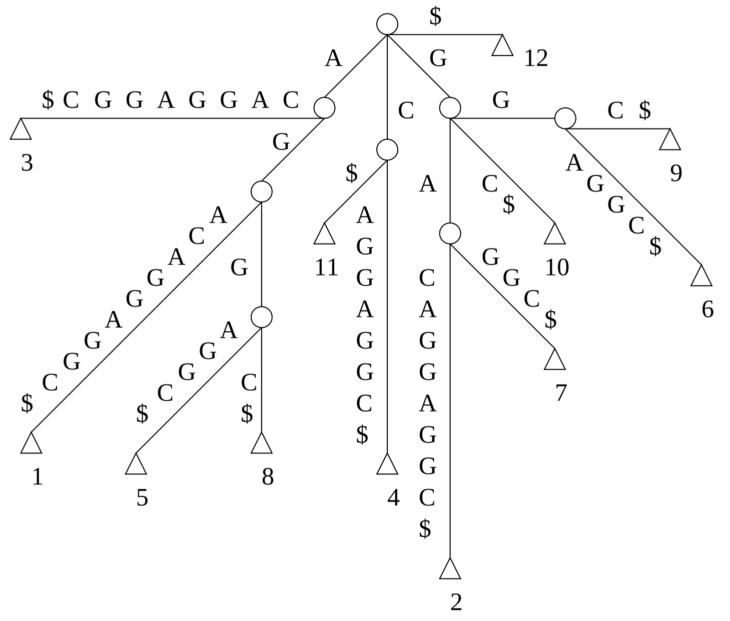


Figure 3.1: Suffix tree for string AGACAGGAGGC$.

by storing at each tree node v a Boolean array, called the $Colors_v$ array (Sagot, 1998), usually implemented as a bit vector. Such array indicates the strings in the input set that contain the suffix labelling the path from the root to the tree node v. As an example, the generalized suffix tree with *Colors* for the same strings S_1=TACTA$ and S_2=CACTCA$, over the DNA alphabet $\Sigma=\{A,C,G,T\}$, is presented in Figure 3.3.

We are not going to present any of the algorithms for the construction of suffix trees. However, it is worthwhile to notice that, by comparing all the linear-time construction algorithms (Weiner, 1973; McCreight, 1976; Ukkonen, 1995), Ukkonen's method (Ukkonen, 1995) is one that uses less space in practice, therefore being the method of choice for most problems requiring the construction of a suffix tree.

Factor tree

Factor trees are a new data structure to index strings, proposed by Allali and Sagot (2004), very similar to suffix trees. This data structure, also called the *at most k-deep factor tree* or *k-factor tree*, indexes the factors of a string whose length does not exceed k, and only those. Indeed, a factor tree is nothing more than a suffix tree pruned at the labels of depth k. We are not going to present the *k-factor tree* construction algorithm here. However, it is worthwhile

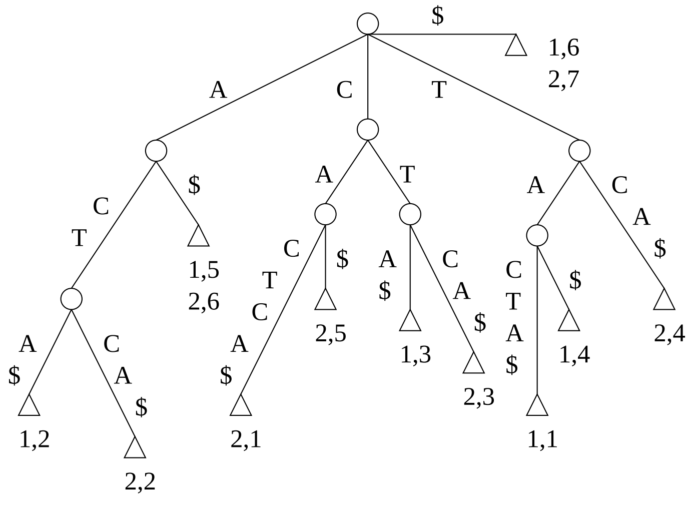


Figure 3.2: Generalized suffix tree for strings TACTA\$ and CACTCA\$.

to notice that it is based on Ukkonen's method (Ukkonen, 1995). As an example, consider the 5-deep factor tree for string $S =$AGACAGGAGGC\$ presented in Figure 3.4. Note that the 5-deep factor tree does not have any leaf with a collapsed start position, since there is no common substring of size 5 in the string $S =$AGACAGGAGGC\$. However, if we consider $k = 3$, the substring AGG occurs twice in the string S, at positions 5 and 8, and we obtain a 3-deep factor tree with collapsed start positions, as depicted in Figure 3.5.

As for suffix trees, the time and space complexities for constructing factor trees are linear in the length of the string (Allali and Sagot, 2004). However, applications such as the extraction of single or structured motifs, where the length of the models to be searched in the suffix tree is limited, can obtain a considerable gain in terms of space and time by the use of factor trees. Compared with a suffix tree, the k-factor tree offers a substantial gain in terms of space complexity for small values of k, as well as a gain in time when used for enumerating all occurrences of a pattern in a text indexed by such a k-factor tree.

To implement the factor tree construction algorithm a new codification is used based on an *Improved Linked List Implementation*, proposed by Kurtz (1999) for suffix trees, called the *illi* coding. Fundamentally, the coding is changed so that it efficiently handles the fact that a leaf in the factor tree may store more than one position. In the factor tree coding, the first occurrence of a leaf added to the factor tree behaves exactly as a leaf added to the suffix tree. However, when trying to insert an already added leaf to the factor tree, a code of the leaf is

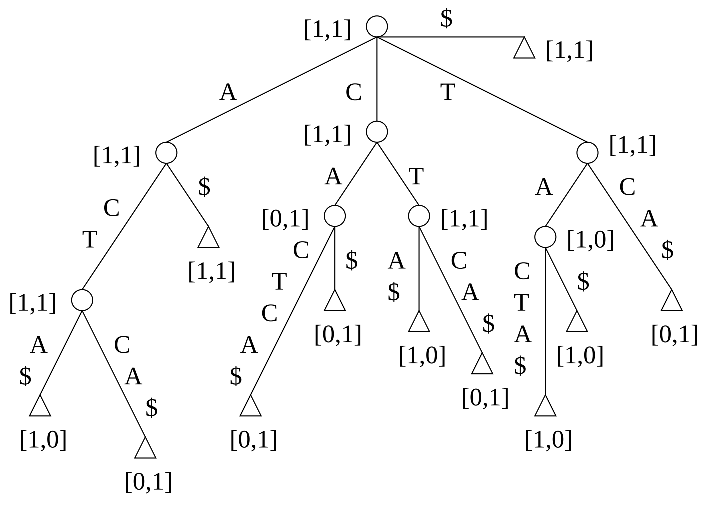


Figure 3.3: Generalized suffix tree with *Colors* for strings TACTA$ and CACTCA$.

also stored but now as a new occurrence of the first one. This new occurrence simply adds a new position to the already existing leaf (c.f. positions 5 and 8 in Figure 3.5). Whenever a new position is added to an already existing leaf we say that the leaf is being updated.

3.2.2 Single motif extraction algorithm

Algorithms for single motif extraction address the extraction of consensus sequences that occur in a subset of the input sequences. In this section we present an algorithm, proposed by Sagot (1998), to extract single motifs (SPELLER). A suffix tree is used to find such motifs in a set of N input sequences over an alphabet Σ. We start by introducing some notation.

Definition 3.2.2 (Model) A *model* is a non-empty string over Σ, that is, it is an element in Σ^+.

A measure of the difference between two sequences of same length over Σ is given by the *Hamming distance*, which is defined as the minimum number of substitutions to transform one sequence into another.

Definition 3.2.3 (Occurence) A model m is said to have an *e-occurrence*, or simply an *occurrence*, in the input sequences, if there is one word u in the input sequences such that the Hamming distance between u and m is less than or equal to e.

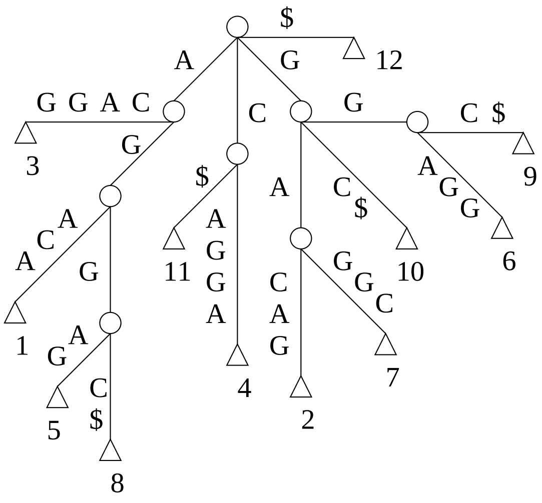


Figure 3.4: At most 5-deep factor tree for string AGACAGGAGGC$.

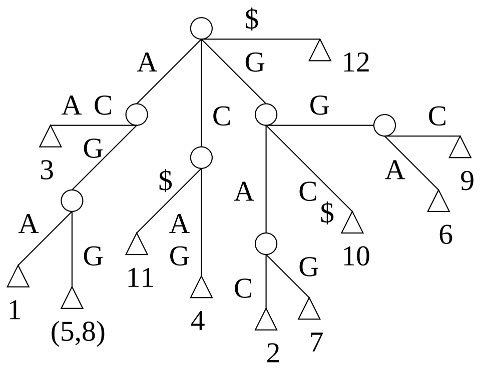


Figure 3.5: At most 3-deep factor tree for string AGACAGGAGGC$.

Definition 3.2.4 (Valid model) A model is said to be a *valid model* if it has an occurrence in at least q input sequences, where q is called the *quorum*.

A valid model is also called a *motif*. In the literature it is common to abuse of notation by using model and motif interchangeably. In this thesis, motif is used when referring to a valid model only.

Definition 3.2.5 (Node-occurence) A *node-occurrence* of a model m is represented by a pair (v, e_v) where v is a tree node and $e_v \leq e$ is the Hamming distance between the label of the path from the root to v and m.

We are now able to describe the algorithm to extract single motifs (Sagot, 1998). At first a suffix tree $\mathcal{T}$ is built for all input sequences. The suffix tree needs to be modified in order to store at each tree node v the $Colors_v$ Boolean array of size N. For the sake of exposition, a *suffix trie* (data structure similar to a suffix tree but with the tree arcs labeled by a single letter) is considered instead of a suffix tree. We shall refer to it as a suffix tree since adapting the algorithm to deal with a compact tree is straightforward. Given a maximum number e of substitutions allowed, it has been shown by Sagot (1998) that extracting all valid k-size models can be done by simultaneously and recursively traversing, in a depth-first way, the lexicographic trie $\mathcal{M}$ of all possible valid models and the suffix tree $\mathcal{T}$ of all input sequences. Observe that, when substitutions are allowed, models that are not represented in the suffix tree may be valid models. In this case, the models that need to be checked for validity are all sequences with Hamming distance at most e from the suffixes of the tree $\mathcal{T}$. Moreover, the lexicographic trie $\mathcal{M}$ is the trie of all these models pruned at the nodes where the quorum is no longer verified. In practice, $\mathcal{M}$ is never built but can be virtually traversed by a more complex traversal over $\mathcal{T}$. Moreover, if no substitutions are allowed and the quorum equals 1 then $\mathcal{M}$ and $\mathcal{T}$ present the same models.

We present the pseudo-code of the algorithm to spell k-size motifs with up to e mismatches in Algorithm 3.7, page 45. The algorithm makes use of the following variables and functions:

- The variable Ext_m, implemented as a bit vector, is a set of symbols by which the model m may be extended at the next step of the algorithm.
- The variable Occ_m is a set of node-occurrences of the model m.
- The variable $Colors_x$ is a Boolean array defined as

$$Colors_x[i] = \begin{cases} 1 & \text{if at least one leaf in the subtree rooted at } x \\ & \text{represents a suffix of the } i\text{-th input sequence } S_i \\ 0 & \text{otherwise.} \end{cases}$$

- The variable $Colors_m$, similarly to $Colors_x$, is a Boolean array defined as

$$Colors_m[i] = \begin{cases} 1 & \text{if } m \text{ occurs in } S_i \\ 0 & \text{otherwise.} \end{cases}$$

- The variable CSS_x, meaning the *color set size* of a node x (Hui, 1992), is the number of different leaf colors in the subtree rooted at x, where a leaf is assigned a color i if it represents a suffix of S_i, that is, $CSS_x = \sum_{i=1}^{N} Colors_x[i]$.

- The variable CSS_m, similarly to CSS_x, is defined as

$$CSS_m = \sum_{i=1}^{N} Colors_m[i].$$

- The variable *minseq* indicates the minimum value of CSS_x for all node-occurrences x of the extended model.

- The variable *maxseq* indicates the sum of the values CSS_x for all node-occurrences x of the extended model.

- The function KeepMotif stores all information concerning valid models for printing later.

The Algorithm 3.7 is called with $(0, \lambda, Occ_\lambda = \{(R, 0)\}, Ext_\lambda)$, where λ is the empty sequence, R is the suffix tree root and

$$Ext_\lambda = \begin{cases} \Sigma & \text{if } e > 0 \\ \text{label of the branches leaving } R & \text{otherwise.} \end{cases}$$

Next, we present the time and space complexity for the Algorithm 3.7. Recall that, N is the number of input sequences and n is the average length of an input sequence. Moreover, n_k is the number of tree nodes at depth k and $\nu(e,k)$ is the number of distinct words that are at a Hamming distance at most e from a k-long word. It is easy to see that the following upper bound for $\nu(e,k)$ holds:

$$\nu(e,k) = \sum_{i=0}^{e} \begin{pmatrix} k \\ i \end{pmatrix} (|\Sigma| - 1)^i \leq k^e |\Sigma|^e.$$

Proposition 3.2.6 (Sagot (1998)) Algorithm 3.7 requires $O(Nn_k\nu(e,k))$ time and $O(N^2n)$ space.

Algorithm 3.7 SPELLER, single motif extraction

```
SPELLER(depth $l$, model $m$, occurrences $Occ_m$, extension $Ext_m$)
1.  if $l = k$ then KeepMotif($m$)
2.  else
3.    for each symbol $\alpha$ in $Ext_m$ do
4.      $maxseq := 0$
5.      $minseq := \infty$
6.      $Colors_{m\alpha} := \overrightarrow{0}$
7.      $Ext_{m\alpha} := \emptyset$
8.      $Occ_{m\alpha} := \emptyset$
9.      for each pair $(x, x_{err})$ in $Occ_m$ do
10.       if there is a branch $b$ leaving node $x$ with a label starting with $\alpha$ then
11.         let $x'$ be the node reached by following branch $b$ from $x$
12.         add to $Occ_{m\alpha}$ the pair $(x', x_{err})$
13.         $maxseq := maxseq + CSS_{x'}$
14.         if $CSS_{x'} < minseq$ then $minseq := CSS_{x'}$
15.         $Colors_{m\alpha} := Colors_{m\alpha} + Colors_x$
16.         $Ext_{m\alpha} := \begin{cases} Ext_{m\alpha} \cup \text{label}_{b'} \text{ for } b' \text{ leaving } x' & \text{if } x_{err} = e \\ \Sigma & \text{otherwise} \end{cases}$
17.       if $x_{err} < e$ then
18.         for each branch $b$ leaving $x$ except for the one labeled with $\alpha$ do
19.           let $x'$ be the node reached by following branch $b$ from $x$
20.           add to $Occ_{m\alpha}$ the pair $(x', x_{err} + 1)$
21.           $maxseq := maxseq + CSS_{x'}$
22.           if $CSS_{x'} < minseq$ then $minseq := CSS_{x'}$
23.           $Colors_{m\alpha} := Colors_{m\alpha} + Colors_x$
24.           $Ext_{m\alpha} := \begin{cases} Ext_{m\alpha} \cup \text{label}_{b'} \text{ for } b' \text{ leaving } x' & \text{if } x_{err} = e - 1 \\ \Sigma & \text{otherwise} \end{cases}$
25.     if $maxseq < q$ then return
26.     else if $minseq \geq q$ or $CSS_{m\alpha} \geq q$ then SPELLER($l + 1, m\alpha, Occ_{m\alpha}, Ext_{m\alpha}$)
```

3.2.3 Structured motif extraction algorithm

Algorithms for structured motif extraction address the extraction of consensus motifs that appear together in a well-ordered and regularly spaced manner. Structured motifs were first introduced by Marsan and Sagot (2000). In this section we present a structured motif extraction algorithm (RISO) proposed by Carvalho, Freitas, Oliveira, and Sagot (2006). For that purpose we first present a data structure, called *box-link*, which is responsible for an exponential time gain over previous structure motif extraction algorithms (Marsan and Sagot, 2000). Its construction from the input sequences is straightforward and it will be omitted from this thesis (we refer the reader to the works of Carvalho, Freitas, Oliveira, and Sagot, 2006; Carvalho, 2004, for further details). Then, we present the algorithm to extract structured motifs using the box-link data structure. Finally, we introduce some extensions to the algorithm and explain how the extracted models are trimmed by statistical significance in order to deal with the enormous number of false positives.

To set up the algorithm to extract structured motifs, we have to introduce some notation. A structured model can be described as an ordered collection of p boxes, a maximum number e of substitutions allowed for each box, and an interval of distance for each pair of consecutive boxes.

Definition 3.2.7 (Structured model) A *structured model* is a pair (m, d) where:

- $m = (m_i)_{1 \leq i \leq p}$ is a p-tuple of single models, denoting the p boxes;
- $d = (d_{min_i}, d_{max_i}, \delta_i)_{1 \leq i \leq p-1}$ is a $(p-1)$-tuple of triplets, denoting the $p-1$ intervals of distance.

The terms $d_{min_i} \leq d_{max_i}$ represent a minimum and maximum allowed distance between consecutive boxes and δ_i an allowed neighbourhood within that distance. The δ_i is omitted when $\delta_i = (d_{max_i} - d_{min_i} + 1)/2$.

Definition 3.2.8 (Valid structured model) A structured model (m, d) is said to be a *valid structured model* if for all $1 \leq i \leq p-1$ and for all occurrences u_i of m_i, there exist occurrences $u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_p$ of the single motifs $m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_p$ such that:

- $u_1, \ldots, u_p$ belong to the same input sequence;
- there exists d_i, with $d_{\text{min}_i} + \delta_i \leq d_i \leq d_{\text{max}_i} - \delta_i$, such that the distance between the end position of u_i and the start position of u_{i+1} in the sequence is in $[d_i - \delta_i, d_i + \delta_i]$;
- d_i is the same for p-tuples of occurrences present in at least q distinct sequences.

As for single motifs, a valid structured model is also called a *structured motif.*

Box-link data structure

A box-link stores the information needed to jump from box to box in a structured model. Its name comes from the fact that it links all p boxes of a possibly valid structured model. There are two main advantages in the use of this data structure. First and foremost, the information needed to jump from box to box when searching for structured models is memorized and can be quickly accessed. Second, it capitalizes on the use of factor trees, since there is no need to compute the suffix tree below the maximum size of the boxes of the structured models being extracted.

Loosely speaking, a box-link is a tuple of tree nodes, corresponding to jumps on the factor tree from box to box in a structured model. To illustrate the general idea behind box-links, suppose we have the input sequence AAACCCCCGGGGGT and we are extracting structured models with $p = 3$ boxes of the same size $k = 3$, and the same distance $d = 2$ between them. Under these conditions, there are only two box-links for the given input sequence, since there are at most two structured models. Box-links are illustrated in Figure 3.6. Note that only a 3-deep factor tree is needed to work out this problem.

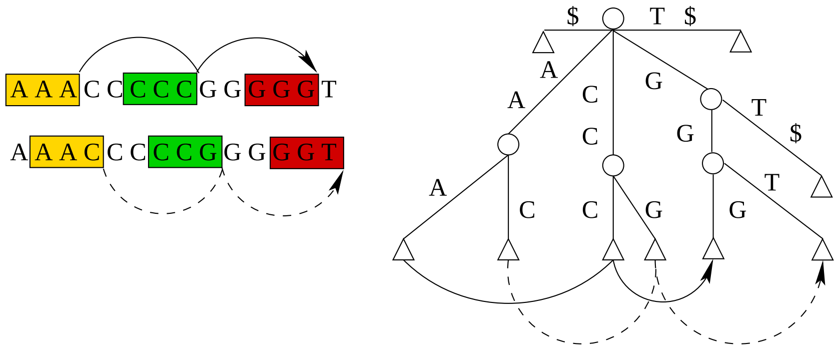


Figure 3.6: A general idea of box-links.

For the sake of simplicity we assume that all p boxes of a structured model are of the same size k with distances between them ranging over the interval $[d_{min}, d_{max}]$. Rigorously, a box-link can be defined as follows:

Definition 3.2.9 (Box-link) Let L_k be the set of leaves at depth k of a k-factor tree $\mathcal{T}$ for a string S and L_k^i denote all possible i-tuples over L_k. A *box-link of size* i, with $1 \leq i < p$, is a $(i+1)$-tuple in L^{i+1} such that there is a substring S' of S where: (i) the length of S' is

$ik+(i-1)d$; (ii) the k-length substring of S' ending at position $jk+(j-1)d$, with $1 \le j \le i$, is the path in $\mathcal{T}$ spelled from the root to the j-th leaf of the box-link tuple.

To store the information relative to which input sequence the box-link refers to, box-links are endowed with a Boolean array, similar to the one associated with tree nodes, defined as:

$$Colors_{B_l}[i] = \begin{cases} 1 & \text{if } B_l \text{ links boxes of the } i\text{-th input sequence} \\ 0 & \text{otherwise.} \end{cases}$$

Jumping in the factor tree using box-links

In this section, we describe the algorithm to extract structured motifs, using box-links as a fundamental data structure.

For clarity of exposition, we consider only structured motifs with two boxes. Moreover, we assume that each box has the same size k, distanced by some value in the interval $[d_{min}, d_{max}]$, and the same maximum allowed error e per box. The RISO algorithm using box-links works as follows. At first, a factor tree $\mathcal{T}$ is built, up to the level k, for all input sequences. The factor tree is then modified to store at each node the *Colors* array, and box-links are added to the leaves of the factor tree, also endowed with its Boolean array. After this pre-processing phase the extraction begins. The structured motif extraction algorithm starts by extracting single motifs of length k, one at a time, as described in Section 3.2.2. For each node-occurrence v of a first box m_1 (Figure 3.7a), a jump is made on the factor tree using the box-links at v (Figure 3.7b). In this jump, the content of the Boolean *Colors* array of the box-links is grabbed. At the target node, the grabbed information is used to temporarily and partially modify the factor tree (Figure 3.7c). The extraction of the second box m_2 of a potentially valid structured model then proceeds in the same way (Figure 3.7d). Once the operation of extracting all possible valid motifs $\langle(m_1, m_2),(d_{min}, d_{max})\rangle$ has ended, the factor tree is restored to its previous state. The construction of another single motif m_1 follows, and the whole process unwinds in a recursive way until all valid structured motifs are extracted.

Suppose we have all p boxes of the same size k with distances between them over the interval $[d_{min}, d_{max}]$. The pseudo-code for the extraction is presented in Algorithm 3.8.

The RISO algorithm makes use of the following functions:

- The function UpdateTree updates the Boolean arrays from the nodes in $NextEnds$ to the root in the following way: if nodes $\overline{z}$ and $\widehat{z}$ have the same parent z, then $Colors_z =$

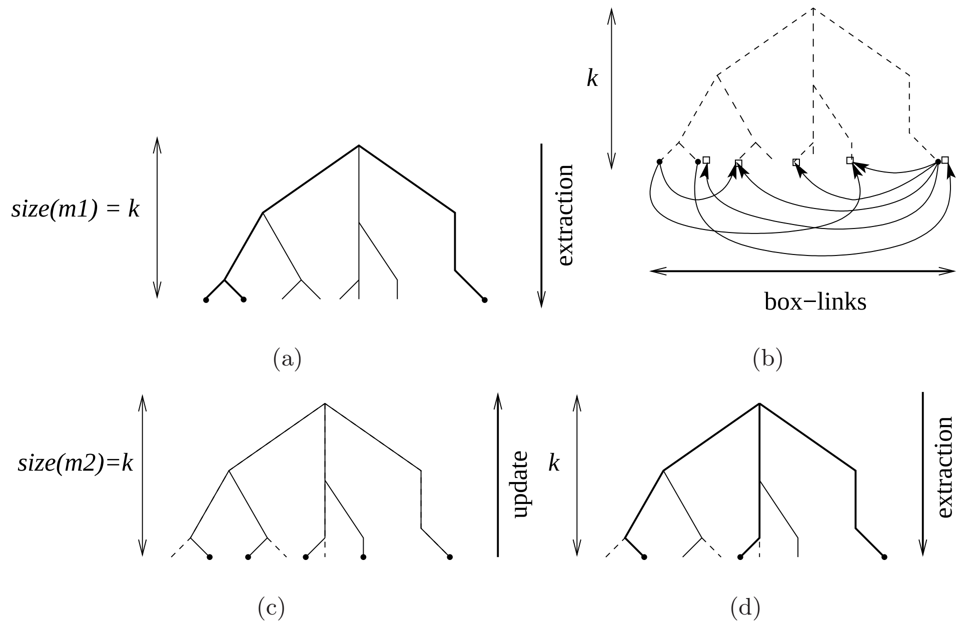


Figure 3.7: Extracting structured motifs following box-links.

Algorithm 3.8 RISO, structured motif extraction using box-links

```
RISO(model $m$, box $i$)
 1. for each node-occurrence $v$ of $m$ do
 2.   for each box-link $B_l(v, z)$ do
 3.     put $z$ in $L(i)$
 4.     if first time $z$ is reached then
 5.       $Colors_z := \overrightarrow{0}$
 6.       put $z$ in $NextEnds$
 7.     $Colors_z := Colors_z + Colors_{B_l(v,z)}$
 8. UpdateTree($\mathcal{T}, NextEnds$)
 9. for each motif $m_i$ obtained by SPELLER traversing $\mathcal{T}$ do
10.   if $i < p$ then RISO($m = m_1 \dots m_i, i + 1$)
11.   else KeepMotif($m = \langle(m_1, \dots, m_p), (d_{min}, d_{max})\rangle$)
12. RestoreTree($\mathcal{T}, L(i)$)
```

$Colors_{\overline{z}}$ + $Colors_{\widehat{z}}$. Any arc from the root that does not have a node in $NextEnds$ is not part of the updated tree, nor are the subtrees rooted at its node in $NextEnds$.

- The function RestoreTree restores the Boolean arrays from the nodes in $L(i)$ to the root in the following way: if nodes $\overline{z}$ and $\widehat{z}$ have the same parent z, then $Colors_z = Colors_{\overline{z}}$ + $Colors_{\widehat{z}}$. Any arc from the root is part of the restored tree.

- The function KeepMotif stores all information concerning valid motifs.

Next, we present the time and space complexity of RISO for the case when $d_{max} = d_{min} = d$. We take an upper bound for the total number of $(p-1)$-size box-links defined by $b_p(k,d) = \min\{n_k^p, n_{pk+(p-1)d}\}$.

Proposition 3.2.10 (Carvalho, Freitas, Oliveira, and Sagot (2006)) Algorithm 3.8 takes $O(Ns_p(k,d)\nu^p(e,k) + Nb_p(k,d)\nu^{p-1}(e,k))$ time and $O(Nb_p(k,d) + Npn_k)$ space.

Extending the algorithm

The RISO algorithm assumed that all single motifs m_i of a structured motif (m, d) have a unique fixed size k, the same substitution rate e and identical values for (d_{min}, d_{max}). The original paper that established the algorithm (Carvalho et al., 2006; Carvalho, 2004) presents extensions to handle boxes with variable length k_i, variable substitution rate e_i and variable intervals of distance (d_{min_i}, d_{max_i}). It also shows how to deal with restricted intervals of unknown limits $(d_{min_i}, d_{max_i}, \delta_i)$. Moreover, it emphasizes how local and global constraints can be introduced. In particular, besides fixing a maximum substitution rate for each box of a structured motif, it also establishes a maximum substitution rate e_{global} for the whole structured motif. Such a global rate allows to consider, in a limited way, possible correlations between boxes. Another presented local (or global) constraint imposes the frequency of one or more nucleotides in a box (or among all boxes) to be below or above a certain threshold.

Measuring statistical significance

Once all structured motifs have been extracted, they are classified according to their statistical significance, in an attempt to give them some biological pertinence. There is not in the literature a fully satisfactory method for evaluating such significance. Marsan and Sagot

(2000) and Carvalho, Freitas, Oliveira, and Sagot (2006) used a data shuffling approach (Karlin et al., 1989) to evaluate the significance of a structured model. In order to obtain the statistical significance of the models found, a χ^2 test, with one degree of freedom, is performed on two contingency tables: one table expressing what was observed, and another corresponding to what is expected under the null hypothesis (Press et al., 1993). To derive the values in the contingency table for the null hypothesis several random shufflings are performed preserving the k-mer frequency distribution of the input sequences (a k-mer is a substring of length k). Both the number of shufflings and k are values given by the user (in general, 100 shufflings are considered conserving di- or tri- nucleotides). With this process, the probability of getting the models observed under the null hypothesis is estimated. Another type of statistics, based on a Z-score, was also tried by the authors (Carvalho, Freitas, Oliveira, and Sagot, 2006; Carvalho, 2004; Marsan and Sagot, 2000).

Availability

The source code and binaries of the RISO algorithm are freely available in the author web site.[4] There is also an on-line version of RISO as a running component of the YEASTRACT-DISCOVERER tool.[5] For this tool RISO was augmented to consider reverse strands and the statistical analysis was reshaped. For more details see the joint work of Monteiro, Mendes, Teixeira, d'Orey, Tenreiro, Mira, Pais, Francisco, Carvalho, Lourenço, Sá-Correia, Oliveira, and Freitas (2008).

[4]RISO source code and binaries are available at `http://kdbio.inesc-id.pt/ asmc/software/riso.html`.
[5]An on-line version of RISO is available at `http://www.yeastract.com/discoverer/riso.php`.

Part II

Motif representation

Chapter 4

$\hat{\mathrm{f}}\mathrm{CLL}$: Factorized conditional log-likelihood

Bayesian networks have been widely used in the context of classification (Su and Zhang, 2006; Grossman and Domingos, 2004; Friedman et al., 1997). Seminal works in learning Bayesian network classifiers attempted to optimize the *log-likelihood* (LL) of the entire data. This has been pointed out to be the reason why some elaborate classifiers underperform the much simpler naive Bayes classifier (Domingos and Pazzani, 1997; Friedman et al., 1997). Indeed, the goal of classification is to maximize the *conditional log-likelihood* (CLL) of the class variable given the attributes and so maximizing LL, or a score thereof, may result in some suboptimal choice during the learning process.

At the light of the previous discussion two distinct learning methodologies appeared: *generative learning* which optimizes the LL, or a penalized variant, of the entire data being generated by the model, including the class variable; and, *discriminative learning* which aims at maximizing the CLL, focusing on correctly discriminating between classes. The latter approach has received considerable attention in recent years carrying a problem of computational nature: CLL does not decompose over the network structure, and so there is no closed-form equation for choosing the optimal parameters for the CLL scoring criterion. This issue led Friedman et al. (1997) to bring up a new open question: are there heuristic approaches that allow efficient discriminative learning of Bayesian network classifiers? During the past years different approximations were proposed which decomposed the problem into two simpler approaches: (i) search for the Bayesian-network structure that maximizes CLL;

(ii) considering a Bayesian-network structure fixed in advance, compute the parameters that maximize CLL.

The first work along these lines was proposed by Greiner and Zhou (2002). They introduced a discriminative parameter learning algorithm, called *Extended Logistic Regression* (ELR) algorithm, that resorts to a gradient descent optimization method to find maximum-CLL parameters. The ELR algorithm applies to an arbitrary Bayesian-network structure, usually computed by a generative learning method, and then parameters are set by maximum-CLL, using a gradient descent optimization method. The rationale for this approach is that discriminatively-learned parameters would presumably be advantageous, when compared to generatively-trained ones, in the presence of an incorrect Bayesian-network structure. Greiner and Zhou (2002) have shown that such approach, although, computationally expensive, is feasible and worthwhile. Based on these results, Su et al. (2008) tried to overcome the ELR computational cost and proposed an alternative discriminative parameter learning algorithm, called *DFE algorithm*, which exhibits the same accuracy as the ELR algorithm but that it is considerably more efficient. Notwithstanding, despite current empirical evidence supporting the effectiveness of the DFE algorithm, its theoretical nature remains unknown and a deeper understanding of the employed technique is needed.

Full structure and parameter learning with the ELR algorithm is a burdensome task. Indeed, employing Greiner and Zhou (2002) procedure to discriminative structure learning would require a new gradient descent for each candidate network at each search step, turning the method computationally unfeasible. Moreover, even in the context of discriminative parameter learning, ELR is not guaranteed to find optimal-CLL parameters. Fortunately, there are also some positive results that favor ELR. Discriminative parameter learning in a naive Bayes network structure is known to be equivalent to a logistic regression problem. Roos et al. (2005) showed that this equivalence still holds for network structures that satisfy a certain graph-theoretic property. Such property holds for naive Bayes but also for more complex structures such as *tree augmented naive Bayes* (TAN) networks, devised by Friedman et al. (1997), and others. Their result implies that for such networks CLL cannot have local maxima. As a consequence, the global maximum can be found by local optimization methods, such as ELR. In fact, Greiner and Zhou (2002) verified this result empirically by concluding that their ELR algorithm is beneficial when combined with generatively-trained

TAN classifiers.

In an opposite direction Grossman and Domingos (2004) proposed to choose network structures by maximizing CLL while approximating parameters by their maximum-LL. The algorithm employed by the authors, called *BNC algorithm*, is similar to the hill-climbing algorithm of Heckerman et al. (1995), except that it uses CLL as the primary objective function. The BNC algorithm starts from an empty network, and, at each step, considers adding each possible new edge and deleting or reversing each current edge. Since CLL does not decompose over the network structure, the full network is then scored using the CLL scoring criterion, considering that the parameters have been set to their maximum-LL values. The reasoning behind this approach is that computing maximum-LL parameter estimates is extremely fast, and, for an optimal network structure, they are asymptotically equivalent to the maximum-CLL ones. To a limited extent, full structure and parameter optimization for CLL were also studied by the authors. However, experiments revealed that full optimization did not produce better results than those obtained by the much simpler previous approach.

The methods proposed by Greiner and Zhou (2002), Su et al. (2008) and Grossman and Domingos (2004) are a great step towards discriminative learning. Nevertheless, they leave out full exploitation of both structure and parameter learning algorithms that maximize CLL, in its possible extent, since CLL is not decomposable over the network structure. Our contribution works in this line and is based on an approximation, for binary classification tasks, to CLL, using the least squares' method. Focusing on learning the structure of augmented naive Bayes classifiers from complete data, we derived a decomposable scoring criterion, called *approximated conditional log-likelihood* (aCLL), that minimizes the expected squared error with respect to CLL. The proposed score is the *minimum variance unbiased* (MVU) approximation to CLL, which means that the expected value of the difference between CLL and aCLL is zero and, among the unbiased approximations is the one with minimum variance. Unfortunately, the parameters that maximize aCLL are unknown. A possibly effective solution, for full discriminative learning, would be choosing Bayesian-network structures by maximizing aCLL while setting optimal-CLL parameters by using the ELR algorithm. However, aCLL was devised having in mind efficiency, reason why we proposed a decomposable approximation to CLL, so the use of ELR ends clashing with our goal.

For the sake of computational efficiency, we considered aCLL with parameters set by the *observed frequency estimates* (OFE) hoping that discriminatively-structured networks work well even in the presence of non-optimal parameter setting. The resulting score, called âCLL, brings up a problem, however. In fact, âCLL is not well-behaved under OFE as it might be indeterminate for some datasets. Therefore, we devised a well-behaved approximation to aCLL for which the parameters are maximized by OFE. The score obtained was called *factorized conditional log-likelihood* ($\hat{\text{f}}$CLL). Contrarily to âCLL, $\hat{\text{f}}$CLL is score equivalent and it corresponds to adding to the LL a summand that incorporates *interaction information* between a node, its parents and the class variable. It seems clear that interaction information is an interesting quantity for classification. It prefers choosing parents for a node that increase the information about the node when joined with the class to parents that increase the information about the node, but explain the behavior of the node without any contribution of the class. Actually, since interaction information is not always positive, $\hat{\text{f}}$CLL does not prefer, in general, more complex structures to simpler ones. Interaction information has been referred to as *explaining away residual* (EAR) by Bilmes (2000) and has been extensively discussed in the machine learning community (Cover and Thomas, 2006; Pearl, 1988)). The EAR metric has already been used by Pernkopf and Bilmes (2005) in the context of Bayesian network classifiers for global discriminative learning. As far as we are concerned it is the first time this measures appears in local discriminative learning.

To gauge the performance of the proposed criteria in classification tasks, we compare them with several popular classifiers, namely, *tree augmented naive Bayes* (TAN), *greedy hill-climbing* (GHC), C4.5, *k-nearest neighbor* (k-NN), *support vector machine* (SVM) and *logistic regression* (LogR). On a large suite of benchmark datasets from the UCI repository, $\hat{\text{f}}$CLL-trained classifiers outperform, with high significance level, their generatively-trained counterparts, as well as C4.5, k-NN and LogR classifiers. Moreover, $\hat{\text{f}}$CLL-optimal classifiers are comparable with ELR induced ones, as well as SVMs (with linear, polynomial and radial basis function kernels). The advantage of $\hat{\text{f}}$CLL with respect to these latter classifiers is that it is computationally as efficient as the LL scoring criterion, and considerably more efficient than ELR and SVMs.

This chapter is organized as follows. In Section 4.1 we update notation introduced in Section 3.1.1, in page 21, to classification tasks. In Section 4.2 we provide background material in Bayesian network classifiers along with generative and discriminative classification. In Section 4.3 we present our scoring criteria followed by experimental results and their analysis in Section 4.4. Finally, in Appendix A we provide further explanation about the work presented herein.

4.1 Notation

In the context of augmented naive Bayes models the class variable is a special attribute. To avoid misunderstandings, when devising new scoring criteria for classification tasks, the class variable needs to be decoupled from the other parent attributes. Herein, we set up a notation that takes this observation into account. We propose a notation as accordant as possible to the well established notation presented in Section 3.1.1 (page 21), for defining scoring criteria for learning Bayesian networks (see, for instance, de Campos (2006)).

We consider that the class variable C ranges over s values $z_1, \ldots, z_s$. We recall that the number of states of the finite random variable X_i is r_i and that the parents of X_i are denoted by Π_{X_i}. The parents of X_i without the class variable are denoted by $\Pi^*_{X_i} = \Pi_{X_i} \setminus \{C\}$. We denote the number of possible configurations of the parent set $\Pi^*_{X_i}$ by q^*_i. Hence,

$$q^*_i = \prod_{X_j \in \Pi^*_{X_i}} r_j.$$

The j-th configuration of $\Pi^*_{X_i}$ is represented by w^*_{ij}, with $1 \leq j \leq q^*_i$. Similarly to the general case, local distributions are determined by the corresponding parameters

$$\begin{aligned}
&P(C = c) = \theta_c, \\
&P(X_i = x_{ik} \mid \Pi^*_{X_i} = w^*_{ij}, C = z_c) = \theta_{ijck}.
\end{aligned}$$

We denote by N_{ijck} the number of instances in the data T where the attribute X_i takes its k-th value x_{ik}, the attributes in $\Pi^*_{X_i}$ take their j-th configuration w^*_{ij} and the class variable C takes its c-th value z_c. Moreover, N_{ij*k} denotes the number of instances in the data T where the attribute X_i takes its k-th value x_{ik} and the attributes in $\Pi^*_{X_i}$ take their j-th configuration w^*_{ij}, disregarding the value of the class variable, that is,

$$N_{ij*k} = \sum_{c=1}^{s} N_{ijck}.$$

In addition, N_{ijc} is the number of instances in the data T where the attributes in $\Pi^*_{X_i}$ take their j-th configuration w^*_{ij} and the class variable C takes its c-th value z_c, that is,

$$N_{ijc} = \sum_{k=1}^{r_i} N_{ijck}.$$

Furthermore, N_{ij*} is the number of instances in the data T where the attributes in $\Pi^*_{X_i}$ take their j-th configuration w^*_{ij} regardless of the class variable C, that is,

$$N_{ij*} = \sum_{k=1}^{r_i}\sum_{c=1}^{s} N_{ijck}.$$

Similarly, N_c is the number of instances in the data T where the class variable C takes its c-th value z_c, that is,

$$N_c = \sum_{k=1}^{r_i}\sum_{j=1}^{q^*_i} N_{ijck},$$

for any $1 \leq i \leq n$. Finally, recall that $T = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$. In classification tasks the t-th instance of T is given by $\mathbf{y}_t = (y^1_t, \ldots, y^n_t, c_t)$, where c_t corresponds to the value of the class variable. Note that the total number of instances in the data T is N.

The Table 4.1 resumes the notation needed in this chapter.

4.2 Generative vs discriminative learning

Having in mind the notation above, we are ready to briefly discuss the two paradigms for learning Bayesian network classifiers: generative and discriminative. *Generative learning* reduces to maximizing the likelihood of the data, including the class variable. When the structure of the network is fixed in advance this reduces to estimating the parameters θ_{ijck}. In this case, the ML estimates in Equation (3.4) at page 30 become now

$$\hat{\theta}_c = \frac{N_c}{N}, \qquad \text{and} \qquad \hat{\theta}_{ijck} = \frac{N_{ijck}}{N_{ijc}}. \tag{4.1}$$

Therefore, plugging again these parameters into the *log-likelihood* (LL) scoring criterion we obtain:

$$\begin{aligned}\widehat{\text{LL}}(G \mid T) &= \sum_{t=1}^{N} \log(P_B(y^1_t, \ldots, y^n_t, c_t)) \\ &= \sum_{c=1}^{s}\left(\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{c=1}^{s}\sum_{k=1}^{r_i} N_{ijck}\log\left(\frac{N_{ijck}}{N_{ijc}}\right)\right) + N_c \log\left(\frac{N_c}{N}\right). \end{aligned} \tag{4.2}$$

Symbol	**Meaning**
T	training data, $T = \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$
$\mathbf{y}_t$	t-th instance of T, $\mathbf{y}_t = (y_t^1, \dots, y_t^n, c_t)$
N	number of instances in T
B	(augmented naive) Bayesian network classifier
$\mathbf{X}$	set of attributes $X_1, \dots, X_n$ in B, excluding the class variable C
n	number of attributes, excluding the class variable C
x_{ik}	k-th value that the attribute X_i can take
r_i	number of values X_i can take
C	class variable in B
z_c	c-th value that the class variable C can take
s	number of values C can take
Π_{X_i}	parents of X_i in B
$\Pi^*_{X_i}$	parents of X_i in B without the class variable C
q_i^*	number of possible parent configurations of $\Pi^*_{X_i}$
w_{ij}^*	j-th (parent) configuration of $\Pi^*_{X_i}$
N_c	number of of instances in T where $C = z_c$
N_{ij*}	number of instances in T where $\Pi^*_{X_i} = w^*_{ij}$
N_{ijc}	number of instances in T where $\Pi^*_{X_i} = w^*_{ij}$ and $C = z_c$
N_{ij*k}	number of instances in T where $X_i = x_{ik}$ and $\Pi^*_{X_i} = w^*_{ij}$
N_{ijck}	number of instances in T where $X_i = x_{ik}$, $\Pi^*_{X_i} = w^*_{ij}$ and $C = z_c$
$(y_t^1, \dots, y_t^n, 1 - c_t)$	dual of the t-th instance in T (may not occur in T)
U_t	probability of the t-th instance in T
V_t	probability of the dual of the t-th instance in T
P_B	joint distribution of $(\mathbf{X}, C)$ induced by B
$\hat{P}_T$	joint distribution of $(\mathbf{X}, C)$ induced by the OFE parameters

Table 4.1: Definition of terms used in Chapter 4.

The $\widehat{\text{LL}}$ scoring criterion tends to favor complex network structures with many edges since adding an edge never decreases likelihood. This phenomenon leads to *overfitting* which is usually avoided by adding a complexity penalty to the log-likelihood or by restricting the network structure.

On the other hand, *discriminative learning* of Bayesian network classifiers maximizes the conditional likelihood of the data. The reason why this is a form of discriminative learning is that it focus on correctly discriminating between classes by maximizing the probability of obtaining the correct classification. The *conditional log-likelihood* (CLL) scoring criterion can be written as:

$$\text{CLL}(B \mid T) = \sum_{t=1}^{N} \log(P_B(c_t \mid y_t^1, \ldots, y_t^n)). \tag{4.3}$$

Friedman et al. (1997) noticed that, in the context of classification learning problems, the log-likelihood of T for B can be rewritten as:

$$\text{LL}(B \mid T) = \text{CLL}(B \mid T) + \sum_{t=1}^{N} \log(P_B(y_t^1, \ldots, y_t^n)). \tag{4.4}$$

Interestingly, the objective of generative learning is precisely to maximize the whole sum, whereas the goal of discriminative learning consists on maximizing only the first term in (4.4). Friedman et al. (1997) attributed the underperformance of learning methods based on LL to the term $\text{CLL}(B \mid T)$ being potentially much smaller than the second term in Equation (4.4). Unfortunately, CLL does not decompose over the network structure, which seriously hinders structure learning (see Bilmes, 2000; Grossman and Domingos, 2004). Furthermore, there is no closed-form formula for optimal parameter estimates maximizing CLL, and computationally more expensive techniques such as ELR are required (Greiner and Zhou, 2002; Su et al., 2008).

4.3 Developing a new scoring criterion

The above shortcomings of earlier discriminative approaches to learning Bayesian network classifiers, and the CLL criterion in particular, make it natural to explore good approximations to the CLL that are more amenable to efficient optimization. More specifically, we now set out to construct *decomposable* approximations to the CLL scoring criterion.

For simplicity, assume that the class variable is binary, $C = \{0, 1\}$. The conditional probability of the class variable can then be written as

$$P_B(c_t \mid y_t^1, \dots, y_t^n) = \frac{P_B(y_t^1, \dots, y_t^n, c_t)}{P_B(y_t^1, \dots, y_t^n, c_t) + P_B(y_t^1, \dots, y_t^n, 1 - c_t)}. \tag{4.5}$$

For convenience, we denote the two terms in the denominator as

$$U_t = P_B(y_t^1, \dots, y_t^n, c_t) \qquad \text{and} \qquad V_t = P_B(y_t^1, \dots, y_t^n, 1 - c_t), \tag{4.6}$$

so that Equation (4.5) becomes simply

$$P_B(c_t \mid y_t^1, \dots, y_t^n) \;=\; \frac{U_t}{U_t + V_t}. \tag{4.7}$$

We stress that both U_t and V_t depend on B, but for the sake of readability we omit B. Moreover, observe that $(y_t^1, \dots, y_t^n, 1 - c_t)$ may not occur in the dataset T. We call the sample $(y_t^1, \dots, y_t^n, 1 - c_t)$ the *dual* sample of $(y_t^1, \dots, y_t^n, c_t)$ and so, V_t is the probability of observing the dual sample of the t-th instance in T.

For the case of binary classification the log-likelihood (LL), and the conditional log-likelihood (CLL), of T for B has the following form:

$$\begin{aligned} \mathrm{LL}(B \mid T) &= \sum_{t=1}^{N} \log(U_t)\ \text{, and} \qquad (4.8)\\ \mathrm{CLL}(B \mid T) &= \sum_{t=1}^{N} \log\left(\frac{U_t}{U_t + V_t}\right) = \sum_{t=1}^{N} \log(U_t) - \log(U_t + V_t). \end{aligned}$$

Recall that our goal is to derive a decomposable scoring criterion. Unfortunately, even though $\log(U_t)$ decompose over the network structure, $\log(U_t + V_t)$ does not. In order to achieve decomposability we need to determine which expressions involving the logarithm of U_t and V_t would result in a decomposable scoring criterion with a closed-form expression. Despite the overwhelming number of possibilities the properties of the logarithm highly constrain the number of candidate expressions which would result in a decomposable score. To put it another way, from the properties of the logarithm, the only expressions we found involving U_t and V_t that denote a decomposable score are of the form

$$\alpha \log(U_t) + \beta \log(V_t) + \gamma, \tag{4.9}$$

where α, β and γ are real numbers. Therefore, let us consider approximating the log-ratio

$$f(U_t, V_t) = \log\left(\frac{U_t}{U_t + V_t}\right),$$

by functions of the form

$$\hat{f}(U_t, V_t) = \alpha \log(U_t) + \beta \log(V_t) + \gamma,$$

where α, β, and γ are real numbers to be chosen so as to minimize the approximation error. Since the accuracy of the approximation obviously depends on the values of U_t and V_t as well as the constants α, β, and γ, we need to make some assumptions about U_t and V_t in order to determine suitable values of α, β and γ. We explain one possible set of assumptions, which will be seen to lead to a good approximation for a very wide range of U_t and V_t. We emphasize that the role of the assumptions is to aid in arriving at good choices of the constants α, β, and γ, after which we can dispense with the assumptions — they need not, in particular, hold true exactly.

Start by noticing that $R_t = 1 - (U_t + V_t)$ is the probability of observing neither the t-th sample nor its dual, and hence, the triplet (U_t, V_t, R_t) are the parameters of a trinomial distribution. We assume, for the time being, that no knowledge about the values of the parameters (U_t, V_t, R_t) is available. Therefore, it is natural to assume that (U_t, V_t, R_t) follows the uniform Dirichlet distribution, Dirichlet$(1, 1, 1)$, which implies that

$$(U_t, V_t) \sim \text{Uniform}(\Delta^2), \tag{4.10}$$

where $\Delta^2 = \{(x, y) : x + y \leq 1 \text{ and } x, y \geq 0\}$ is the 2-simplex set. However, with a brief reflection on the matter, we can see that such an assumption is actually rather unrealistic. Firstly, by inspecting the total number of possible observed samples, we expect, R_t to be relatively large (close to 1). In fact, U_t and V_t are expected to become exponentially small as the number of attributes grows. Therefore, it is reasonable to assume that

$$U_t, V_t \leq p < \frac{1}{2} < R_t \tag{4.11}$$

for some positive value p. Combining this constraint with the uniformity assumption, Equation (4.10), yields the following assumption, which we will use as a basis for our further analysis.

Assumption 1 There exists a small positive $p < \frac{1}{2}$ such that

$$(U_t, V_t) \sim \text{Uniform}(\Delta^2)|_{U_t, V_t \leq p < \frac{1}{2}} = \text{Uniform}([0, p] \times [0, p]). \tag{4.12}$$

Assumption 1 implies that U_t and V_t are uniform i.i.d. random variables ranging over $[0,p]$ for some (unknown) positive real number $p < \frac{1}{2}$. In Appendix A we give an alternative justification for this assumption.

As we shall see later, we do not need to know the actual value of p. A relevant consequence of this fact is that the envisaged approximation will be robust to the choice of p. Under Assumption 1, we obtain analytically the best fitting that minimizes the average difference between f and $\hat{f}$ by the least squares' method.

Theorem 4.3.1 Under Assumption 1, the values of α, β and γ that minimize the *mean square error* (MSE) of $\hat{f}$ w.r.t. f are given by

$$\begin{aligned}
\alpha &= \frac{\pi^2+6}{24}, && (4.13)\\
\beta &= \frac{\pi^2-18}{24}, \text{ and} && (4.14)\\
\gamma &= \frac{\pi^2}{12\ln(2)} - \left(2 + \frac{(\pi^2-6)\log(p)}{12}\right), && (4.15)
\end{aligned}$$

where $\log(\cdot)$ is the binary logarithm and $\ln(\cdot)$ is the natural logarithm.

Proof: We have that

$$\begin{aligned}
S_p(\alpha,\beta,\gamma) &= \int_0^p\int_0^p \frac{1}{p^2}\left(\log\left(\frac{x}{x+y}\right) - (\alpha\log(x)+\beta\log(y)+\gamma)\right)^2 \mathrm{d}y\,\mathrm{d}x\\
&= \frac{1}{12\ln(2)^2}(-\pi^2(-1+\alpha+\beta)+\\
&\quad 6(2+4\alpha^2+4\beta^2-4\ln(2)-2\gamma\ln(2)+4\ln(2)^2+8\gamma\ln(2)^2+2\gamma^2\ln^2(2)\\
&\quad +\beta(5-4(2+\gamma)\ln(2))+\alpha(1+4\beta-4(2+\gamma)\ln(2)))\\
&\quad -12(\alpha+\beta)(1+2\alpha+2\beta-4\ln(2)-2\gamma\ln(2))\ln(p)+12(\alpha+\beta)^2\ln^2(p)).
\end{aligned}$$

Moreover, $\nabla.S_p = 0$ iff

$$\begin{aligned}
\alpha &= \frac{\pi^2+6}{24},\\
\beta &= \frac{\pi^2-18}{24},\\
\gamma &= \frac{\pi^2}{12\ln(2)} - \left(2 + \frac{(\pi^2-6)\log(p)}{12}\right),
\end{aligned}$$

which coincides exactly with (4.13), (4.14) and (4.15), respectively. Now to show that (4.13), (4.14) and (4.15) define a global minimum, take $\delta = (\alpha \log(p) + \beta \log(p) + \gamma)$ and notice that

$$\begin{aligned}
S_p(\alpha,\beta,\gamma) &= \int_0^p \int_0^p \frac{1}{p^2}\left(\log\left(\frac{x}{x+y}\right) - (\alpha\log(x)+\beta\log(y)+\gamma)\right)^2 \mathrm{d}y\,\mathrm{d}x \\
&= \int_0^1 \int_0^1 \frac{1}{p^2}\left(\log\left(\frac{px}{px+py}\right) - (\alpha\log(px)+\beta\log(py)+\gamma)\right)^2 p^2\mathrm{d}y\,\mathrm{d}x \\
&= \int_0^1 \int_0^1 \left(\log\left(\frac{x}{x+y}\right) - (\alpha\log(x)+\beta\log(y)+(\alpha\log(p)+\beta\log(p)+\gamma))\right)^2 \mathrm{d}y\,\mathrm{d}x \\
&= \int_0^1 \int_0^1 \left(\log\left(\frac{x}{x+y}\right) - (\alpha\log(x)+\beta\log(y)+\delta)\right)^2 \mathrm{d}y\,\mathrm{d}x \\
&= S_1(\alpha,\beta,\delta).
\end{aligned}$$

So, S_p has a minimum at (4.13), (4.14) and (4.15) iff S_1 has a minimum at (4.13), (4.14) and

$$\delta = \frac{\pi^2}{12\ln(2)} - 2. \tag{4.16}$$

The Hessian of S_1 in α, β and δ given by (4.13), (4.14) and (4.16), is

$$\begin{pmatrix} \frac{4}{\ln^2(2)} & \frac{2}{\ln^2(2)} & -\frac{2}{\ln(2)} \\ \frac{2}{\ln^2(2)} & \frac{4}{\ln^2(2)} & -\frac{2}{\ln(2)} \\ -\frac{2}{\ln(2)} & -\frac{2}{\ln(2)} & 2 \end{pmatrix}$$

and its eigenvalues are

$$\begin{aligned}
e_1 &= \tfrac{3+\ln^2(2)+\sqrt{9+2\ln^2(2)+\ln^4(2)}}{\ln^2(2)}, \\
e_2 &= \tfrac{2}{\ln^2(2)}, \\
e_3 &= \tfrac{3+\ln^2(2)-\sqrt{9+2\ln^2(2)+\ln^4(2)}}{\ln^2(2)},
\end{aligned}$$

which are all positive. Thus, S_1 has a local minimum in (α, β, δ) and, consequently, S_p has a local minimum in (α, β, γ). Since $\nabla.S_p$ has only one zero, (α, β, γ) is a global minimum of S_p. □

We then study two desirable properties of $\hat{f}$: unbiasedness and minimum variance approximation. We show that the difference between the expected value of $\hat{f}$ and f is zero, that is, $\hat{f}$ is an *unbiased approximation* of f. Moreover, we show that $\hat{f}$ is the approximation with the lowest variance among unbiased ones.

Theorem 4.3.2 The approximation $\hat{f}$ with α, β, γ defined as in Theorem 4.3.1 is the *minimum variance unbiased* (MVU) approximation of f.

Proof: We have that

$$\int_0^p \int_0^p \frac{1}{p^2}\left(\log\left(\frac{x}{x+y}\right) - (\alpha\log(x) + \beta\log(y) + \gamma)\right) \mathrm{d}y\, \mathrm{d}x = 0$$

for α, β and γ defined as in (4.13), (4.14) and (4.15). Moreover, we have that for unbiased approximations the MSE coincides with the variance. Therefore, the proposed approximation is the one with minimum variance. □

Next, we derive the error of the approximation $\hat{f}$ in the square $[0, p] \times [0, p]$, which, curiously, does not depend on p. We consider

$$\mu = E[f(U_t, V_t)] = \frac{1}{2\ln(2)} - 2 \tag{4.17}$$

which is a negative value since f ranges over $(-\infty, 0]$.

Theorem 4.3.3 The approximation $\hat{f}$ with α, β and γ defined as in Theorem 4.3.1 has *standard error* given by

$$\sigma = \sqrt{\frac{36 + 36\pi^2 - \pi^4}{288\ln^2(2)} - 2} \approx 0.352114 \tag{4.18}$$

and *relative standard error*

$$\frac{\sigma}{|\mu|} \approx 0.275379.$$

Proof: We have that

$$\sqrt{\int_0^p \int_0^p \frac{1}{p^2}\left(\log\left(\frac{x}{x+y}\right) - (\alpha\log(x) + \beta\log(y) + \gamma)\right)^2 \mathrm{d}y\, \mathrm{d}x} = \sqrt{\frac{36 + 36\pi^2 - \pi^4}{288\ln^2(2)} - 2}$$

for α, β and γ defined as in (4.13), (4.14) and (4.15), which concludes the proof. □

Figure 4.1 illustrates the function f as well as its approximation $\hat{f}$ for $(U_t, V_t) \in [0, p] \times [0, p]$ with $p = 0.05$. Both functions diverge (to $-\infty$) as $U_t \to 0$. The latter diverges (to $+\infty$) also when $V_t \to 0$. For the interpretation of different colors in Figure 4.1, see Figure 4.2 (page 68). The approximation error, $f - \hat{f}$ is shown in Figure 4.2. The approximation error represents the difference between the exact value and the approximation given in Theorem 4.3.1. Notice that the error is symmetric in the two arguments, and diverges as $U_t \to 0$ or $V_t \to 0$. For points where neither argument is close to zero, the error is small (close to zero). While the

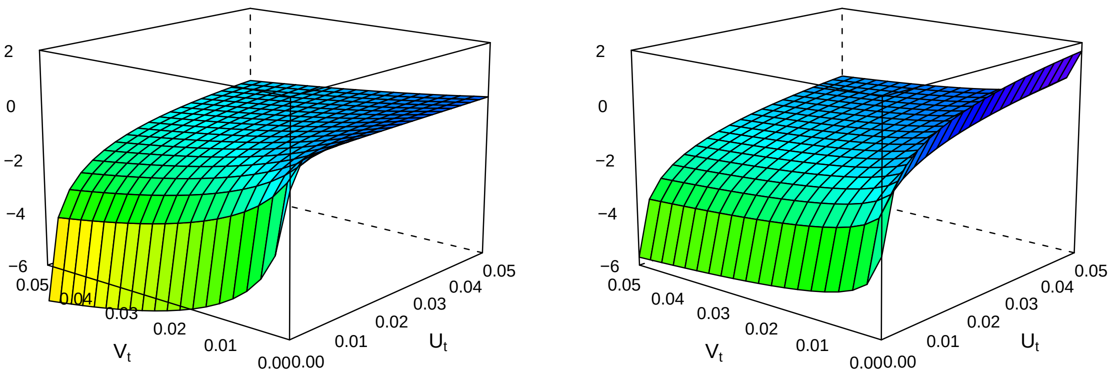


Figure 4.1: Comparison between f (left) and $\hat{f}$ (right).

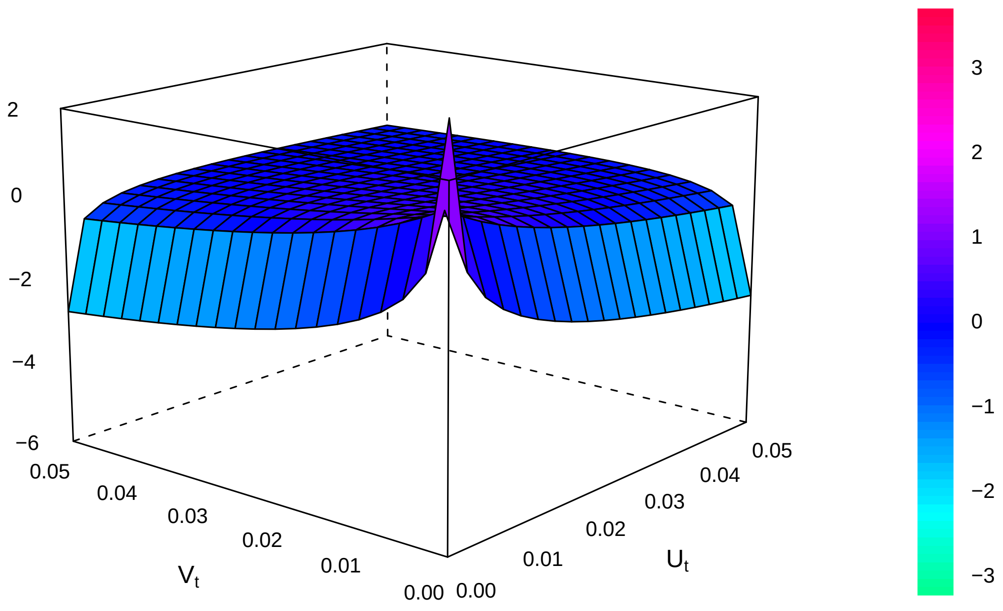


Figure 4.2: Approximation error between f and $\hat{f}$.

properties established in Theorem 4.3.1–4.3.3 are useful, we find it even more important that, as seen in Figure 4.2, the error is close to zero except when either U_t or V_t approaches zero. Moreover, we point out that the choice of $p = 0.05$ used in the figure is not important: having chosen another value would have produced identical graphs except in the scale of the U_t and V_t. In particular, the scale and numerical values on the vertical axis (i.e., in Figure 4.2, the error) would have been precisely the same.

By taking the approximation in Theorem 4.3.1, we have that

$$\begin{aligned}\mathrm{CLL}(B \mid T) = \sum_{t=1}^{N} \log\left(\frac{U_t}{U_t+V_t}\right) &\approx \sum_{t=1}^{N} \alpha \log(U_t) + \beta \log(V_t) + \gamma \\ &= \sum_{t=1}^{N} (\alpha+\beta)\log(U_t) - \beta \log\left(\frac{U_t}{V_t}\right) + \gamma \\ &= (\alpha+\beta)\mathrm{LL}(B \mid T) - \beta \sum_{t=1}^{N} \log\left(\frac{U_t}{V_t}\right) + N\gamma, \qquad (4.19)\end{aligned}$$

where constants α, β and γ are given by Equations (4.13), (4.14) and (4.15), respectively. Since we want to maximize $\mathrm{CLL}(B \mid T)$ we can drop the constant $N\gamma$ in the approximation, as maxima are invariant under monotonous transformations, and so we can just maximize the following formula, which we call the *approximate conditional log-likelihood* (aCLL):

$$\begin{aligned}\mathrm{aCLL}(B \mid T) &= (\alpha+\beta)\mathrm{LL}(B \mid T) - \beta \sum_{t=1}^{N} \log\left(\frac{U_t}{V_t}\right) \\ &= (\alpha+\beta)\,\mathrm{LL}(B \mid T) - \beta \sum_{i=1}^{n} \sum_{j=1}^{q_i^*} \sum_{k=1}^{r_i} \sum_{c=0}^{1} N_{ijck} \log\left(\frac{\theta_{ijck}}{\theta_{ij(1-c)k}}\right) \\ &\quad -\beta \sum_{c=0}^{1} N_c \log\left(\frac{\theta_c}{\theta_{(1-c)}}\right). \qquad (4.20)\end{aligned}$$

At this stage we are ready to understand why the approximation in Theorem 4.3.1 results in a scoring criterion which is robust to the choice of p. Indeed, the fact that $N\gamma$ can be removed from the maximization in (4.19) is most fortunate, as we eliminate the dependency on p. An immediate consequence of this fact is that we do not need to know the actual value of p for a given dataset T. Indeed, the approximation works for any positive $p < \frac{1}{2}$ by just changing the constant γ, which is irrelevant for maximizing.

Another significant concern that deserves some discussion is the analysis of the propagation of the error found in Theorem 4.3.3 when we are summing approximations as in (4.19). In this context it is interesting to notice that errors may cancel, namely, because we are summing i.i.d. samples. Indeed, we have that U_t and V_t are i.i.d. for all $1 \leq t \leq N$, since T is a multinomial sample. Therefore, we conclude that the approximation given by (4.19) is unbiased, its standard error is $\sigma\sqrt{N}$ and its relative standard error is $\frac{\sigma}{\sqrt{N}|\mu|}$, where σ and μ are given by (4.18) and (4.17), respectively. To sum up, the relative standard error decreases proportionally with the square root of N.

We are now in the position of having constructed a *decomposable* approximation of the conditional log-likelihood score that was shown to be very accurate for a wide range of parameters U_t and V_t. Due to the dependency of these parameters on Θ, i.e., the parameters of the Bayesian network B (recall Equation (4.6), page 63), the score still requires that a suitable set of parameters is chosen. Finding the parameters maximizing the approximation is, however, difficult; apparently as difficult as finding parameters maximizing the CLL directly. Therefore, whatever computational advantage is gained by decomposability, it would seem to be dwarfed by the expensive parameter optimization phase.

Furthermore, trying to use the OFE parameters in aCLL may lead to problems since the approximation is undefined at points where either U_t or V_t is zero. To better see why this is the case, substitute the OFE parameters, Equation (4.1), into the aCLL criterion, Equation (4.20), to obtain

$$\begin{aligned}\hat{\mathrm{a}}\mathrm{CLL}(G \mid T) = (\alpha+\beta)\,\widehat{\mathrm{LL}}(G \mid T) - \beta \sum_{i=1}^{n}\sum_{j=1}^{q_i^*}\sum_{k=1}^{r_i}\sum_{c=0}^{1} N_{ijck}\log\left(\frac{N_{ijck}N_{ij(1-c)}}{N_{ijc}N_{ij(1-c)k}}\right)\\ -\beta\sum_{c=0}^{1} N_c \log\left(\frac{N_c}{N_{1-c}}\right). \qquad (4.21)\end{aligned}$$

The problems are associated with the denominator in the second term. In LL and CLL criteria, similar expressions where the denominator may be zero are always eliminated by the OFE parameters since they are always multiplied by zero, see e.g., Equation (4.2), where $N_{ijc} = 0$ implies $N_{ijck} = 0$. However, there is no guarantee that $N_{ij(1-c)k}$ is non-zero even if the factors in the numerator are non-zero, and hence the division by zero may lead to actual indeterminacies. This problem makes âCLL *not well-behaved*, since it has *singularities* that are *infinite discontinuities*.[1] Therefore, depending on the dataset, âCLL might behave well or not. Unfortunately, this problem arises more often than one might expect. The reason for this is that âCLL depends on local counting, determined by the network structure, made over dual samples. Moreover, dual samples might never occur in the data, making the local counting $N_{ij(1-c)k}$ and $N_{ij(1-c)}$ to be zero occasionally. In practice, we found many cases where $\hat{\mathrm{a}}\mathrm{CLL}(B \mid T)$ score was not defined while learning from the UCI datasets. To address this issue we considered the standard solution based on pseudo-counts in which we sum to

[1] By taking $N_{ij(1-c)k}$ to be a positive real number, which indeed we consider when using pseudo-counts, we have that $\lim_{N_{ij(1-c)k}\to 0} \hat{\mathrm{a}}\mathrm{CLL}(B \mid T) = +\infty$ for some dataset T.

each N_{ijck} a fixed number of pseudo-counts. The other quantities N_{ijc}, N_{ij*k} and N_{ij} are updated as expected taking into account this modification. However, it is not obvious which are the values of the pseudo-counts that will give a meaningful smoothing to the score. If the pseudo-counts are very small, the score explodes and behaves badly, and if the pseudo-counts are very high, they smooth too much the data, leading to a poor classifier. A detailed study to control the bad behavior of $\hat{\mathrm{a}}\mathrm{CLL}(B \mid T)$ is not straightforward.

The previous conclusions deserve an insightful discussion we provide next. Start by considering the very first expression of $\mathrm{CLL}(B \mid T)$ for binary classification tasks in (4.9), or simply in (4.7). Somewhat unsurprisingly, CLL tries to discriminate between class labels by examining which instances in the data simultaneously do not occur as dual samples (note that the highest value of (4.7) is attained when $V_t = 0$). Although the lack of dual samples seems extraordinary for classifying, eliciting a zero probability from that can lead to irrecoverable errors. It is a common mistake to assign probability zero to an event that is extremely unlikely, but not impossible (see e.g. Koller and Friedman (2009)). This is precisely the reason why pseudo-counts are so widely used when learning Bayesian networks from data. Regrettably, in our case, even using reasonable pseudo-counts a problem of order of magnitude persists as the second summand in (4.21) dominates the first one in practical cases (find some examples in the supplementary material webpage). This final consideration leads us, in fact, to make an approximation to the second summand that corrects this problem, by ensuring that the first summand and the approximation to the second summand have similar orders of magnitude. Moreover, the relative values of both the approximation and the second summand should be retained, that is, the highest value given by the envisaged approximation should be attained when the second summand has its maximum value (or, more precisely, their derivative should have the same sign).

We conclude this section by examining why aCLL under OFE (or, simply âCLL) is not well-behaved even if by Assumption 1 the expected approximation error is zero. Actually, by assuming discrete-valued attributes the number of all possible datasets is countable. Therefore, the number of distributions that are obtained by OFE is also countable. When we focus on a countable subset of values (in our case, given by the OFE) over a continuous domain (in our case, given by the $\mathrm{Uniform}([0,p] \times [0,p])$) we lose much of the properties that characterize this domain. Consequently, we have to make further assumptions to ensure that the subset preserve the desirable characteristics of the continuos domain. One desideratum is that the envisaged approximation should be well-defined, under OFE, with probability one (or, similarly, the probability of having singularities should be zero). Clearly, âCLL does not fulfill this property. This long discussion is just an introduction to the next section where we set out to resolve these issues by presenting a well-behaving approximation that enables easy optimization of both structure (via decomposability), as well as parameters.

4.3.1 Achieving a well-behaved approximation under OFE

In this section, we address the singularities of aCLL under OFE by constructing an approximation that is well-behaved.

Recall aCLL in Equation (4.20). Given a fixed network structure, the parameters that maximize the first term, $(\alpha+\beta)\text{LL}(B \mid T)$, are given by OFE. However, as observed above, the second term may actually be unbounded due to the appearance of $\theta_{ij(1-c)k}$ in the denominator. In order to obtain a well-behaved score, we must therefore make a further modification to the second term. Our strategy is to ensure that the resulting score is *uniformly bounded* and *maximized by OFE parameters.* The intuition behind this is that we can thus guarantee not only that the score is well-behaved, but also that parameter learning is achieved in a simple and efficient way by using the OFE parameters — solving both of the aforementioned issues with the aCLL score. As it turns out, we can satisfy our goal while still retaining the discriminative nature of the score.

The following result is of importance in what follows.

Theorem 4.3.4 Consider a Bayesian network B whose structure is given by a fixed directed acyclic graph, G. Let $f(B \mid T)$ be a score defined by

$$f(B \mid T) = \sum_{i=1}^{n}\sum_{j=1}^{q_i^*}\sum_{k=1}^{r_i}\sum_{c=0}^{1} N_{ijck}\left(\lambda \log\left(\frac{\theta_{ijck}}{\frac{N_{ijc}}{N_{ij*}}\theta_{ijck} + \frac{N_{ij(1-c)}}{N_{ij*}}\theta_{ij(1-c)k}}\right)\right), \tag{4.22}$$

where λ is an arbitrary positive real value. Then, the parameters Θ that maximize $f(B \mid T)$ are given by the observed frequency estimates (OFE) obtained from G.

For the proof of Theorem 4.3.4 we need to recall Gibb's inequality. This inequality states that the entropy $-\sum_x P(x)\log(P(x))$ of a probability distribution $P(x)$ is less than or equal to its cross entropy with any other probability distribution $Q(x)$.

Lemma 4.3.5 (Gibb's inequality) Let $P(x)$ and $Q(x)$ be two probability distributions over the same domain, then

$$\sum_{x} P(x)\log(Q(x)) \leq \sum_{x} P(x)\log(P(x)).$$

Proof (Theorem 4.3.4): We now take advantage of Gibb's inequality to show that the parameters that maximize the $f(B \mid D)$ are those given by the OFE. Observe that

$$\begin{aligned} f(B \mid D) &= \lambda \sum_{i=1}^{n}\sum_{j=1}^{q_i^*}\sum_{k=1}^{r_i}\sum_{c=0}^{1} N_{ijck}\log\left(\frac{N_{ijc}\theta_{ijck}}{N_{ijc}\theta_{ijck}+N_{ij(1-c)}\theta_{ij(1-c)k}}\right) - \log\left(\frac{N_{ijc}}{N_{ij*}}\right) \\ &= K + \lambda \sum_{i=1}^{n}\sum_{j=1}^{q_i^*}\sum_{k=1}^{r_i} N_{ij*k}\sum_{c=0}^{1}\frac{N_{ijck}}{N_{ij*k}}\log\left(\frac{N_{ijc}\theta_{ijck}}{N_{ijc}\theta_{ijck}+N_{ij(1-c)}\theta_{ij(1-c)k}}\right), \qquad (4.23) \end{aligned}$$

where K is a constant that does not depend on the parameters θ_{ijck}, and therefore, can be ignored. Moreover, if we take the OFE for the parameters, we have

$$\hat{\theta}_{ijck} = \frac{N_{ijkc}}{N_{ijc}} \quad \text{and} \quad \hat{\theta}_{ij(1-c)k} = \frac{N_{ijk(1-c)}}{N_{ij(1-c)}}.$$

By plugging the OFE estimates in (4.23) we obtain

$$\begin{aligned} \hat{f}(G \mid D) &= K + \lambda \sum_{i=1}^{n}\sum_{j=1}^{q_i^*}\sum_{k=1}^{r_i} N_{ij*c}\sum_{c=0}^{1}\frac{N_{ijck}}{N_{ij*k}}\log\left(\frac{N_{ijc}\frac{N_{ijck}}{N_{ijc}}}{N_{ijc}\frac{N_{ijck}}{N_{ijc}}+N_{ij(1-c)}\frac{N_{ij(1-c)k}}{N_{ij(1-c)}}}\right) \qquad (4.24) \\ &= K + \lambda \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ij*k}\sum_{c=0}^{1}\frac{N_{ijck}}{N_{ij*k}}\log\left(\frac{N_{ijck}}{N_{ij*k}}\right). \end{aligned}$$

According to Gibb's inequality, this is the maximum value that $f(B \mid D)$ can attain, and therefore, the parameters that maximize $f(B \mid D)$ are those given by the OFE. □

The Theorem 4.3.4 implies that by replacing the second term in (4.20) by (a non-negative multiple of) $f(B \mid T)$ in Equation (4.22), we get a criterion where the first and second terms are maximized by the OFE parameters. We will now proceed to determine a suitable value for the parameter λ appearing in Equation (4.22).

To clarify the analysis, we introduce the following short-hand notations:

$$\begin{aligned} A_1 &= N_{ijc}\theta_{ijck}, & A_2 &= N_{ijc}, \\ B_1 &= N_{ij(1-c)}\theta_{ij(1-c)k}, & B_2 &= N_{ij(1-c)}. \end{aligned} \qquad (4.25)$$

With simple algebra, we can rewrite the logarithm in the second term of Equation (4.20) using the above notations as

$$\begin{aligned}
\log\left(\frac{\theta_{ijck}}{\theta_{ij(1-c)k}}\right) &= \log\left(\frac{N_{ijc}N_{ij(1-c)}}{N_{ijc}N_{ij(1-c)}}\times\frac{\theta_{ijck}}{\theta_{ij(1-c)k}}\right)\\
&= \log\left(\frac{N_{ij(1-c)}}{N_{ijc}}\times\frac{N_{ijc}\theta_{ijck}}{N_{ij(1-c)}\theta_{ij(1-c)k}}\right)\\
&= \log\left(\frac{N_{ijc}\theta_{ijck}}{N_{ij(1-c)}\theta_{ij(1-c)k}}\right)-\log\left(\frac{N_{ijc}}{N_{ij(1-c)}}\right)\\
&= \log\left(\frac{A_1}{B_1}\right)-\log\left(\frac{A_2}{B_2}\right). \qquad (4.26)
\end{aligned}$$

Similarly, the logarithm in (4.22) becomes

$$\begin{aligned}
&\lambda\log\left(\frac{\theta_{ijck}}{\frac{N_{ijc}}{N_{ij*}}\theta_{ijck}+\frac{N_{ij(1-c)}}{N_{ij*}}\theta_{ij(1-c)k}}\right)\\
&=\lambda\log\left(\frac{N_{ijc}N_{ij*}}{N_{ijc}N_{ij*}}\times\frac{\theta_{ijck}}{\frac{N_{ijc}}{N_{ij*}}\theta_{ijck}+\frac{N_{ij(1-c)}}{N_{ij*}}\theta_{ij(1-c)k}}\right)+\rho-\rho\\
&=\lambda\log\left(\frac{N_{ij*}}{N_{ijc}}\times\frac{N_{ijc}\theta_{ijck}}{N_{ijc}\theta_{ijck}+N_{ij(1-c)}\theta_{ij(1-c)k}}\right)+\rho-\rho\\
&=\lambda\log\left(\frac{N_{ijc}\theta_{ijck}}{N_{ijc}\theta_{ijck}+N_{ij(1-c)}\theta_{ij(1-c)k}}\right)+\rho-\lambda\log\left(\frac{N_{ijc}}{N_{ij*}}\right)-\rho\\
&=\lambda\log\left(\frac{N_{ijc}\theta_{ijck}}{N_{ijc}\theta_{ijck}+N_{ij(1-c)}\theta_{ij(1-c)k}}\right)+\rho-\lambda\log\left(\frac{N_{ijc}}{N_{ijc}+N_{ij(1-c)}}\right)-\rho\\
&=\lambda\log\left(\frac{A_1}{A_1+B_1}\right)+\rho-\lambda\log\left(\frac{A_2}{A_2+B_2}\right)-\rho, \qquad (4.27)
\end{aligned}$$

where we used $N_{ij*}=N_{ijc}+N_{ij(1-c)}$; we have introduced the constant ρ that was added and subtracted without changing the value of the expression for a reason that will become clear shortly. By comparing Equation (4.26) and Equation (4.27), it can be seen that the latter is obtained from the former by replacing expressions of the form $\log(\frac{A}{B})$ by expressions of the form $\lambda\log(\frac{A}{A+B})+\rho$.

We can simplify the two-variable approximation to a single variable one by taking $W=\frac{A}{A+B}$. In this case we have that $\frac{A}{B}=\frac{W}{1-W}$, and so we propose to apply once again the least squares method to approximate the function

$$g(W)=\log\left(\frac{W}{1-W}\right)$$

by

$$\hat{g}(W)=\lambda\log(W)+\rho.$$

The role of the constant ρ is simply to translate the approximate function to better match the target $g(W)$.

As in the previous approximation, here too it is necessary to make assumptions about the values of A and B (and thus W), in order to find suitable values for the parameters λ and ρ. Again, we stress that the sole purpose of the assumption is to guide in the choice of the parameters.

As both A_1, A_2, B_1, and B_2 in Equation (4.25) are all non-negative, the ratio $W_i = \frac{A_i}{A_i+B_i}$ is always between zero and one, for both $i \in \{1, 2\}$, and hence it is natural to make the straightforward assumption that W_1 and W_2 are uniformly distributed along the unit interval. This gives us the following assumption.

Assumption 2 We assume that

$$\begin{aligned}
\frac{N_{ijc}\theta_{ijck}}{N_{ijc}\theta_{ijck} + N_{ij(1-c)}\theta_{ij(1-c)k}} &\sim \text{Uniform}(0,1), \quad \text{and} \\
\frac{N_{ijc}}{N_{ijc} + N_{ij(1-c)}} &\sim \text{Uniform}(0,1).
\end{aligned}$$

Herein, it is worthwhile noticing that although the previous assumption was meant to hold for general parameters, in practice, we know in this case that OFE will be used. Hence, Assumption 2 reduces to

$$\frac{N_{ijck}}{N_{ij*k}} \sim \text{Uniform}(0,1), \quad \text{and} \quad \frac{N_{ijc}}{N_{ij*}} \sim \text{Uniform}(0,1).$$

Under this assumption, the mean square error of the approximation can be minimized analytically, yielding the following solution.

Theorem 4.3.6 Under Assumption 2, the values of λ and ρ that minimize the mean square error (MSE) of $\hat{g}$ w.r.t. g are given by

$$\lambda = \frac{\pi^2}{6}, \text{ and} \tag{4.28}$$

$$\rho = \frac{\pi^2}{6\ln(2)}. \tag{4.29}$$

Proof: We have that

$$\begin{aligned}
S(\lambda, \rho) &= \int_0^1 \left(\log\left(\frac{x}{1-x}\right) - (\lambda \log(x) + \rho)\right)^2 \mathrm{d}x \\
&= \frac{6\lambda^2 + \pi^2 + 3\rho^2 \ln^2(2) - \lambda\left(\pi^2 + 6\rho\ln(2)\right)}{3\ln^2(2)}.
\end{aligned}$$

Moreover $\nabla . S = 0$ iff

$$\begin{aligned} \lambda &= \frac{\pi^2}{6}, \\ \rho &= \frac{\pi^2}{6\ln(2)}, \end{aligned}$$

which coincides with (4.28) and (4.29), respectively. The Hessian of S when $\nabla . S = 0$ is

$$\begin{pmatrix} \frac{4}{\ln^2(2)} & -\frac{2}{\ln(2)}, \\ -\frac{2}{\ln(2)} & 2 \end{pmatrix}$$

with eigenvalues

$$\frac{2 + \ln^2(2) \pm \sqrt{4 + \ln^4(2)}}{\ln^2(2)}$$

which are both positive. So, since there is only one zero, (λ, ρ) is a global minimum. □

Theorem 4.3.7 The approximation $\hat{g}$ with λ and ρ defined as in Theorem 4.3.6 is the minimum variance unbiased (MVU) approximation of g.

Proof: We have that

$$\int_0^1 \left(\log\left(\frac{x}{1-x}\right) - (\lambda \log(x) + \rho) \right) \mathrm{d}x = 0,$$

for λ and ρ defined as in (4.28) and (4.29). Moreover, we have that for unbiased approximations the MSE coincides with the variance. Therefore, the proposed approximation is the one with minimum variance. □

In order to get an idea of the accuracy of the approximation $\hat{g}$, consider the graph of $\log\left(\frac{w}{1-w}\right)$ and $\lambda \log(w) + \rho$ in Figure 4.3. It may appear problematic that the approximation gets worse as w tends to one. However this is actually unavoidable since that is precisely where âCLL diverges, and our goal is to obtain a criterion that is uniformly bounded.

To wrap up, we first rewrite the logarithm of the second term in Equation (4.20) using formula (4.26), and then apply the above approximation to both terms to get

$$\log\left(\frac{\theta_{ijck}}{\theta_{ij(1-c)k}}\right) \approx \lambda \log\left(\frac{N_{ijc}\theta_{ijck}}{N_{ijc}\theta_{ijck} + N_{ij(1-c)}\theta_{ij(1-c)k}}\right) + \rho - \lambda \log\left(\frac{N_{ijc}}{N_{ij*}}\right) - \rho, \tag{4.30}$$

where ρ cancels out. A similar analysis can be applied to rewrite the logarithm of the third term in Equation (4.20) leading to

$$\log\left(\frac{\theta_c}{\theta_{(1-c)}}\right) = \log\left(\frac{\theta_c}{1-\theta_c}\right) \approx \lambda \log(\theta_c) + \rho. \tag{4.31}$$

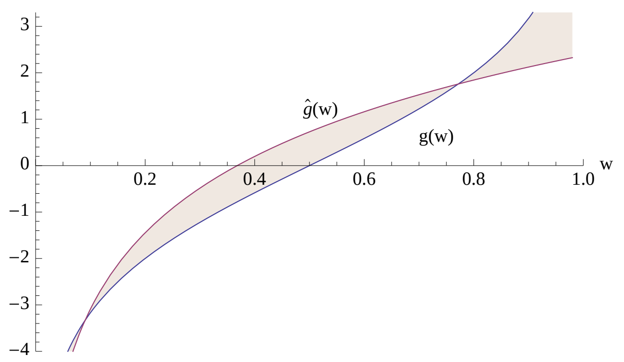


Figure 4.3: Plot of g and $\hat{g}$.

Plugging the approximations of Equations (4.30) and (4.31) into Equation (4.20) gives us finally the *factorized conditional log-likelihood* (fCLL) score:

$$\begin{aligned}\mathrm{fCLL}(B \mid T) &= (\alpha+\beta)\mathrm{LL}(B \mid T)\\ &- \beta\lambda \sum_{i=1}^{n}\sum_{j=1}^{q_i^*}\sum_{k=1}^{r_i}\sum_{c=0}^{1} N_{ijck}\left(\log\left(\frac{N_{ijc}\theta_{ijck}}{N_{ijc}\theta_{ijck}+N_{ij(1-c)}\theta_{ij(1-c)k}}\right) - \log\left(\frac{N_{ijc}}{N_{ij*}}\right)\right)\\ &- \beta\lambda\sum_{c=0}^{1} N_c \log(\theta_c) - \beta N\rho.\end{aligned} \tag{4.32}$$

Observe that the third term of Equation (4.32) is such that

$$-\beta\lambda\sum_{c=0}^{1} N_c\log(\theta_c) = -\beta\lambda N\sum_{c=0}^{1}\frac{N_c}{N}\log(\theta_c), \tag{4.33}$$

and, since $\beta < 0$, by Gibbs inequality (see Lemma 4.3.5 in Appendix A at page 73) the parameters that maximize Equation (4.33) are given by the OFE, that is, $\hat{\theta}_c = \frac{N_c}{N}$. Therefore, by Theorem 4.3.4 (page 72), given a fixed structure, the maximizing parameters of fCLL are easily obtained as OFE. Moreover, the fCLL score is clearly decomposable.

As a final step, we plug in the OFE parameters, Equation (4.1), into the fCLL criterion, Equation (4.32), to obtain

$$\begin{aligned}\hat{\mathrm{f}}\mathrm{CLL}(G \mid T) &= (\alpha+\beta)\widehat{\mathrm{LL}}(B \mid T)\\ &- \beta\lambda\sum_{i=1}^{n}\sum_{j=1}^{q_i^*}\sum_{k=1}^{r_i}\sum_{c=0}^{1} N_{ijck}\left(\log\left(\frac{N_{ijck}}{N_{ij*k}}\right) - \log\left(\frac{N_{ijc}}{N_{ij*}}\right)\right)\\ &- \sum_{c=0}^{1} N_c\beta\lambda\log\left(\frac{N_c}{N}\right) - \beta N\rho,\end{aligned} \tag{4.34}$$

where we also use the OFE parameters in the log-likelihood $\widehat{\text{LL}}$. Observe that we can drop the last two terms in Equation (4.34) as they become constants for a given dataset.

4.3.2 Information-theoretical interpretation

Before we present empirical results illustrating the behavior of the proposed scoring criteria, we point out that the $\hat{\text{f}}$CLL criterion has an interesting information-theoretic interpretation based on *interaction information*. We will first rewrite LL in terms of conditional mutual information, and then, similarly, rewrite the second term of $\hat{\text{f}}$CLL in Equation (4.34) in terms of interaction information.

As Friedman et al. (1997) point out, the local contribution of the i-th variable to $\text{LL}(B \mid T)$ (recall Equation (4.2)) is given by

$$\begin{aligned} N \sum_{j=1}^{q_i^*} \sum_{c=0}^{1} \sum_{k=1}^{r_i} \frac{N_{ijck}}{N} \log \left(\frac{N_{ijck}}{N_{ijc}} \right) &= -N H_{\hat{P}_T}(X_i \mid \Pi^*_{X_i}, C) \\ &= -N H_{\hat{P}_T}(X_i \mid C) + N I_{\hat{P}_T}(X_i \,;\, \Pi^*_{X_i} \mid C), \qquad (4.35) \end{aligned}$$

where $H_{\hat{P}_T}(X_i \mid \ldots)$ denotes the *conditional entropy*, and $I_{\hat{P}_T}(X_i \,;\, \Pi^*_{X_i} \mid C)$ denotes the *conditional mutual information*, see Cover and Thomas (2006). The subscript $\hat{P}_T$ indicates that the information theoretic quantities are evaluated under the joint distribution $\hat{P}_T$ of $(\mathbf{X}, C)$ induced by the OFE parameters.

Since the first term on the right-hand side of (4.35) does not depend on $\Pi^*_{X_i}$, finding the parents of X_i that maximize $\text{LL}(B \mid T)$ is equivalent to choosing the parents that maximize the second term, $N I_{\hat{P}_T}(X_i \,;\, \Pi^*_{X_i} \mid C)$, which measures the information that $\Pi^*_{X_i}$ provides about X_i when the value of C is known.

Let us now turn to the second term of the $\hat{\text{f}}$CLL score in Equation (4.34). The contribution of the i-th variable in it can also be expressed in information theoretic terms as follows:

$$\begin{aligned} -\beta\lambda N \Big(H_{\hat{P}_T}(C \mid X_i, \Pi^*_{X_i}) - H_{\hat{P}_T}(C \mid \Pi^*_{X_i}) \Big) & \\ = \beta\lambda N I_{\hat{P}_T}(C \,;\, X_i \mid \Pi^*_{X_i}) & \\ = \beta\lambda N \Big(I_{\hat{P}_T}(C \,;\, X_i \,;\, \Pi^*_{X_i}) + I_{\hat{P}_T}(C \,;\, X_i)) \Big), & \qquad (4.36) \end{aligned}$$

where $I_{\hat{P}_T}(C \,;\, X_I \,;\, \Pi^*_{X_i})$ denotes the *interaction information* (McGill, 1954), or the *"co-information"* (Bell, 2003); for a review on the history and use of interaction information in machine learning and statistics, see Jakulin (2005).

Since $I_{\hat{P}_T}(X_i\,;C)$ in Equation (4.36) does not depend on $\Pi^*_{X_i}$, finding the parents of X_i that maximize the sum amounts to maximizing the interaction information. This is intuitive, since the interaction information measures the increase — or the decrease, as it can also be negative — of the mutual information between X_i and C when the parent set $\Pi^*_{X_i}$ is included in the model.

All said, the $\hat{\text{f}}$CLL criterion can be written as

$$\hat{\text{f}}\text{CLL}(G \mid T) = \sum_{i=1}^{n} (\alpha+\beta) N I_{\hat{P}_T}(X_i\,; \Pi^*_{X_i} \mid C) - \beta\lambda N I_{\hat{P}_T}(C\,; X_i\,; \Pi^*_{X_i}) + \textit{const}, \tag{4.37}$$

where *const* is a constant independent of the network structure and can thus be omitted. To get a concrete idea of the trade-off between the first two terms, the numerical values of the constants can be evaluated to obtain

$$\hat{\text{f}}\text{CLL}(G \mid T) \approx \sum_{i=1}^{n} 0.322\, N I_{\hat{P}_T}(X_i\,; \Pi^*_{X_i} \mid C) + 0.557 N\, I_{\hat{P}_T}(C\,; X_i\,; \Pi^*_{X_i}) + \textit{const}. \tag{4.38}$$

Normalizing the weights shows that the first term that determines the behavior of the LL criterion, Equation (4.35), has proportional weight of approximately 36.7 percent, while the second term that gives $\hat{\text{f}}$CLL criterion its discriminative nature has the weight 63.3 percent. The particular linear combination of the two terms in Equation (4.38) brings out the question what would happen in only one of the terms was retained, or equivalently, if one of the weights was set to zero. As mentioned above, the first term corresponds to the LL criterion, and hence, setting the weight of the second term to zero would reduce the criterion to LL. We also experimented with a criterion where only the second term is retained but this was observed to yield poor results; for details, see the additional material at the fCLL web page.[2]

In addition to the insight provided by the information-theoretic interpretation of $\hat{\text{f}}$CLL, it also provides a practically most useful corollary: the $\hat{\text{f}}$CLL criterion is score equivalent. Recall that a scoring criterion is said to be *score equivalent* if it assigns the same score to all network structures encoding the same independence assumptions, see Definition 3.1.9 (page 29) or (Verma and Pearl, 1990; Chickering, 2002; Yang and Chang, 2002; de Campos, 2006).

Theorem 4.3.8 The $\hat{\text{f}}$CLL criterion is score equivalent for augmented naive Bayes classifiers.

Proof: By Theorem 2 in Chickering (1995), it is enough to show that for graphs G_1 and G_2 differing only on reversing one covered edge, we have that $\hat{\text{f}}\text{CLL}(G_1 \mid T) = \hat{\text{f}}\text{CLL}(G_2 \mid T)$.

[2] `http://kdbio.inesc-id.pt/∼asmc/software/fCLL.html`

Assume that $X \to Y$ occurs in G_1 and $Y \to X$ occurs in G_2 and that $X \to Y$ is covered, that is, $\Pi_Y^{G_1} = \Pi_X^{G_1} \cup \{X\}$. Since we are only dealing with augment naive Bayes classifiers, X and Y are different from C and so we also have $\Pi_Y^{*G_1} = \Pi_X^{*G_1} \cup \{X\}$. Moreover, take G_0 to be the graph G_1 without the edge $X \to Y$ (which is the same as graph G_2 without the edge $Y \to X$). Then, we have that

$$\Pi_X^{*G_0} = \Pi_Y^{*G_0} = \Pi^{*G_0}$$

and, moreover, the following equalities hold:

$$\begin{array}{ll} \Pi_X^{*G_1} = \Pi^{*G_0}; & \Pi_Y^{*G_2} = \Pi^{*G_0}; \\ \Pi_Y^{*G_1} = \Pi^{*G_0} \cup \{X\}; & \Pi_X^{*G_2} = \Pi^{*G_0} \cup \{Y\}. \end{array}$$

Since $\hat{\text{f}}\text{CLL}$ is a local scoring criterion, $\hat{\text{f}}\text{CLL}(G_1 \mid T)$ can be computed from $\hat{\text{f}}\text{CLL}(G_0 \mid T)$ taking only into account the difference in the contribution of node Y. In this case, by Equation (4.37), it follows that

$$\begin{aligned} \hat{\text{f}}\text{CLL}(G_1 \mid T) &= \hat{\text{f}}\text{CLL}(G_0 \mid T) - ((\alpha+\beta) I_{\hat{P}_T}(Y;\Pi^{*G_0}) - \lambda\beta I_{\hat{P}_T}(Y;\Pi^{*G_0};C)) + \\ &\qquad +((\alpha+\beta) I_{\hat{P}_T}(Y;\Pi_Y^{*G_1}) - \lambda\beta I_{\hat{P}_T}(Y;\Pi_Y^{*G_1};C)) \\ &= \hat{\text{f}}\text{CLL}(G_0 \mid T) + (\alpha+\beta)(I_{\hat{P}_T}(Y;\Pi^{*G_0} \cup \{X\}) - I_{\hat{P}_T}(Y;\Pi^{*G_0})) + \\ &\qquad -\lambda\beta(I_{\hat{P}_T}(Y;\Pi^{*G_0} \cup \{X\};C)) - I_{\hat{P}_T}(Y;\Pi^{*G_0};C)) \end{aligned}$$

and, similarly, we have that

$$\begin{aligned} \hat{\text{f}}\text{CLL}(G_2 \mid T) &= \hat{\text{f}}\text{CLL}(G_0 \mid T) + (\alpha+\beta)(I_{\hat{P}_T}(X;\Pi^{*G_0} \cup \{Y\}) - I_{\hat{P}_T}(X;\Pi^{*G_0})) + \\ &\qquad -\lambda\beta(I_{\hat{P}_T}(X;\Pi^{*G_0} \cup \{Y\};C) - I_{\hat{P}_T}(X;\Pi^{*G_0};C)). \end{aligned}$$

To show that $\hat{\text{f}}\text{CLL}(G_1 \mid T) = \hat{\text{f}}\text{CLL}(G_2 \mid T)$ it suffices to prove that

$$I_{\hat{P}_T}(Y;\Pi^{*G_0} \cup \{X\}) - I_{\hat{P}_T}(Y;\Pi^{*G_0}) = I_{\hat{P}_T}(X;\Pi^{*G_0} \cup \{Y\}) - I_{\hat{P}_T}(X;\Pi^{*G_0}) \tag{4.39}$$

and that

$$I_{\hat{P}_T}(Y;\Pi^{*G_0} \cup \{X\};C) - I_{\hat{P}_T}(Y;\Pi^{*G_0};C) = I_{\hat{P}_T}(X;\Pi^{*G_0} \cup \{Y\};C)) - I_{\hat{P}_T}(X;\Pi^{*G_0};C). \tag{4.40}$$

We start by showing (4.39). In this case, by definition of mutual information, we have that

$$\begin{aligned} I_{\hat{P}_T}(Y;\Pi^{*G_0} \cup \{X\}) - I_{\hat{P}_T}(Y;\Pi^{*G_0}) &= H_{\hat{P}_T}(Y) + H_{\hat{P}_T}(\Pi^{*G_0} \cup \{X\}) - H_{\hat{P}_T}(\Pi^{*G_0} \cup \{X,Y\}) + \\ &\qquad -H_{\hat{P}_T}(Y) - H_{\hat{P}_T}(\Pi^{*G_0}) + H_{\hat{P}_T}(\Pi^{*G_0} \cup \{Y\}) \\ &= -H_{\hat{P}_T}(\Pi^{*G_0}) + H_{\hat{P}_T}(\Pi^{*G_0} \cup \{X\}) + H_{\hat{P}_T}(\Pi^{*G_0} \cup \{Y\}) + \\ &\qquad -H_{\hat{P}_T}(\Pi^{*G_0} \cup \{X,Y\}) \\ &= I_{\hat{P}_T}(X;\Pi^{*G_0} \cup \{Y\}) - I_{\hat{P}_T}(X;\Pi^{*G_0}). \end{aligned}$$

Finally, each addend in (4.40) is, by definition, given by

$$\begin{aligned}
I_{\hat{P}_T}(Y;\Pi^{*G_0}\cup\{X\};C) &= I_{\hat{P}_T}(Y;\Pi^{*G_0}\cup\{X\}\mid C)-\underbrace{I_{\hat{P}_T}(Y;\Pi^{*G_0}\cup\{X\})}_{A}\\
I_{\hat{P}_T}(Y;\Pi^{*G_0};C) &= I_{\hat{P}_T}(Y;\Pi^{*G_0}\mid C)-\underbrace{I_{\hat{P}_T}(Y;\Pi^{*G_0})}_{B}\\
I_{\hat{P}_T}(X;\Pi^{*G_0}\cup\{Y\};C) &= I_{\hat{P}_T}(X;\Pi^{*G_0}\cup\{Y\}\mid C)-\underbrace{I_{\hat{P}_T}(X;\Pi^{*G_0}\cup\{Y\})}_{C}\\
I_{\hat{P}_T}(X;\Pi^{*G_0};C) &= I_{\hat{P}_T}(X;\Pi^{*G_0}\mid C)-\underbrace{I_{\hat{P}_T}(X;\Pi^{*G_0})}_{T}
\end{aligned}$$

and since by (4.39) we know that $A-B=C-D$, for obtaining equality (4.40) it is enough to prove that

$$I_{\hat{P}_T}(Y;\Pi^{*G_0}\cup\{X\}\mid C)-I_{\hat{P}_T}(Y;\Pi^{*G_0}\mid C)=I_{\hat{P}_T}(X;\Pi^{*G_0}\cup\{Y\}\mid C)-I_{\hat{P}_T}(X;\Pi^{*G_0}\mid C).$$

We conclude the proof by noticing that, by definition of conditional mutual information, we have

$$\begin{aligned}
I_{\hat{P}_T}(Y;\Pi^{*G_0}\cup\{X\}\mid C) &- I_{\hat{P}_T}(Y;\Pi^{*G_0}\mid C)=\\
&= H_{\hat{P}_T}(Y\mid C)+H_{\hat{P}_T}(\Pi^{*G_0}\cup\{X\}\mid C)-H_{\hat{P}_T}(\Pi^{*G_0}\cup\{X,Y\}\mid C)+\\
&\quad -H_{\hat{P}_T}(Y\mid C)-H_{\hat{P}_T}(\Pi^{*G_0}\mid C)+H_{\hat{P}_T}(\Pi^{*G_0}\cup\{Y\}\mid C)\\
&= -H_{\hat{P}_T}(\Pi^{*G_0}\mid C)+H_{\hat{P}_T}(\Pi^{*G_0}\cup\{X\}\mid C)+H_{\hat{P}_T}(\Pi^{*G_0}\cup\{Y\}\mid C)+\\
&\quad -H_{\hat{P}_T}(\Pi^{*G_0}\cup\{X,Y\}\mid C)\\
&= I_{\hat{P}_T}(X;\Pi^{*G_0}\cup\{Y\}\mid C)-I_{\hat{P}_T}(X;\Pi^{*G_0}\mid C). \qquad \square
\end{aligned}$$

The practical utility of the above result is due to the fact that it enables the use of powerful algorithms, such as the tree-learning method by Chow and Liu (1968), in learning TAN classifiers.

4.3.3 Beyond binary classification

Although âCLL and $\hat{\text{f}}$CLL scoring criteria were devised having in mind binary classification tasks, their application in multi-classification problems is straightforward. For the case of $\hat{\text{f}}$CLL, the expression (4.34) does not even require any computation based on dual observations. Hence, it can be trivially adapted for non-binary classification tasks. On the other hand, the score âCLL described in (4.21) takes into account dual observations. So, for multi-classification problems, we considered $N_{ij(1-c)k}=N_{ijc}-N_{ijck}$ and $N_{ij(1-c)}=N_{ij}-N_{ijc}$.

Finally, we point out that despite being derived under the augmented naive Bayes model, the $\hat{\text{f}}$CLL score can be readily applied to models where the class variable is *not* a parent of some of the attributes. Hence, we can use it as a criterion for learning more general structures. The empirical results below demonstrate that this indeed leads to good classifiers.

4.4 Experimental results

We implemented the $\hat{\text{f}}$CLL scoring criterion on top of the WEKA java package (Hall et al., 2009). Unfortunately, this open-source package does not provide an implementation of the TAN classifier for non-score-equivalent scoring functions, such as the $\hat{\text{a}}$CLL scoring criterion. This kind of metrics requires Edmonds' algorithm to build a maximal directed spanning tree (see Edmonds, 1967; Lawler, 1976) instead of an undirected one as in Chow-Liu algorithm. Edmonds' algorithm had already been implemented by Carvalho et al. (2007) in Mathematica 7.0 on top of the Combinatorica package (Pemmaraju and Skiena, 2003). Hence, we decided to do a preliminary implementation of $\hat{\text{a}}$CLL in this Mathematica package. The source code, jointly with the datasets used in the experiments, can be found at the fCLL web page.[3]

We evaluated the performance of $\hat{\text{a}}$CLL and $\hat{\text{f}}$CLL scoring criteria in classification tasks comparing them with state-of-the-art classifiers. We performed our evaluation on the same 25 benchmark datasets used by Friedman et al. (1997). These include 23 datasets from the UCI repository of Newman et al. (1998) and two artificial datasets, corral and mofn, designed by Kohavi and John (1997), to evaluate methods for feature subset selection. A description of the datasets is presented in Table 4.2. The continuous-valued attributes in the datasets were discretized in a supervised manner using the entropy-based method suggested by Fayyad and Irani (1993). For this task we used the WEKA package.[4] Moreover, instances with missing values were removed from the datasets.

The classifiers used in the experiments were:

- GHC2: Greedy hill climber classifier with up to 2 parents.
- TAN: Tree augmented naive Bayes classifier.

[3] `http://kdbio.inesc-id.pt/~asmc/software/fCLL.html`

[4] Supervised discretization were performed via `weka.filters.supervised.attribute.Discretize`, with default parameters. This discretization improved the accuracy of all classifiers used in the experiments, including those not requiring discretization (referred to as C4.5, k-NN, SVM's and LogR).

	Dataset	**Features**	**Classes**	**Train**	**Test**
1	australian	15	2	690	CV-5
2	breast	10	2	683	CV-5
3	chess	37	2	2130	1066
4	cleve	14	2	296	CV-5
5	corral	7	2	128	CV-5
6	crx	16	2	653	CV-5
7	diabetes	9	2	768	CV-5
8	flare	11	2	1066	CV-5
9	german	21	2	1000	CV-5
10	glass	10	7	214	CV-5
11	glass2	10	2	163	CV-5
12	heart	14	2	270	CV-5
13	hepatitis	20	2	80	CV-5
14	iris	5	3	150	CV-5
15	letter	17	26	15000	5000
16	lymphography	19	4	148	CV-5
17	mofn-3-7-10	11	2	300	1024
18	pima	9	2	768	CV-5
19	satimage	37	6	4435	2000
20	segment	20	7	1540	770
21	shuttle-small	10	7	3866	1934
22	soybean-large	36	19	562	CV-5
23	vehicle	19	4	846	CV-5
24	vote	17	2	435	CV-5
25	waveform-21	22	3	300	4700

Table 4.2: Description of datasets used in the experiments.

- C4.5: C4.5 classifier.
- k-NN: k-nearest neighbor classifier, with $k = 1, 3, 5$.
- SVM: Support vector machine with linear kernel.
- SVM2: Support vector machine with polynomial kernel of degree 2.
- SVMG: Support vector machine with Gaussian or *radial basis function* (RBF) kernel.
- LogR: Logistic regression.

Bayesian network-based classifiers (GHC2 and TAN) were evaluated in different flavors, depending on the scoring criterion and the estimator used to learn the structure and the parameters. Each variant along with the implementation used in the experiments is described in Table 4.3. Other state-of-the-art classifiers (C4.5, k-NN, SVMs and LogR) used in the experiments are described along side with the respective implementations in Table 4.4. In all implementations default parameters were used, except for those given parenthetically and those described in the next 2 paragraphs.

Classifier	**Struct.**	**Param.**	**Implementation**
GHC2	LL	OFE	`HillClimber` (P=2) implementation from WEKA
GHC2	f̂CLL	OFE	`HillClimber` (P=2) implementation from WEKA
TAN	LL	OFE	`TAN` implementation from WEKA
TAN	LL	ELR	`TAN` implementation from Greiner and Zhou (2002)
TAN	âCLL	OFE	`TAN` implementation from Carvalho et al. (2007)
TAN	f̂CLL	OFE	`TAN` implementation from WEKA

Table 4.3: Bayesian network-based classifiers used in the experiments.

Classifier	**Implementation**
C4.5	`J48` implementation from WEKA
1-NN	`IBk` (K=1) implementation from WEKA
3-NN	`IBk` (K=3) implementation from WEKA
5-NN	`IBk` (K=5) implementation from WEKA
SVM	`SMO` implementation from WEKA
SVM2	`SMO` with `PolyKernel` (E=2) implementation from WEKA
SVMG	`SMO` with `RBFKernel` implementation from WEKA
LogR	`Logistic` implementation from WEKA

Table 4.4: Other state-of-the-art classifiers used in the experiments.

Excluding TAN classifiers obtained with the ELR method, we improved the performance of TAN, as well as of GHC2, classifiers using *Dirichlet priors* (see Heckerman et al. (1995)) to smooth the network parameters. Friedman et al. (1997) found this procedure to be particularly important in small datasets where the estimation of the conditional probabilities, given the parent attributes plus the class variable, is unreliable. We achieve this purpose in WEKA by setting the *alpha* parameter of the OFE to 0.5. In practice, this is the default value for this parameter in WEKA, and the value for which we obtained the highest average accuracy among all classifiers. The same methodology was carried out in the TAN implementation of the Mathematica package. Moreover, for discriminative parameter learning with ELR, the parameters are initialized to the values obtained by the OFE. The gradient descent parameter optimization is terminated using *cross tuning* as suggested in Greiner et al. (2005).

Concerning SVM models, we used three different kernels: (i) a linear kernel of the form $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$; (ii) a polynomial kernel of the form $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^2$; and (iii) a RBF kernel of the form $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2)$. Following the canon of the literature (see Hsu et al., 2003), we used a *grid-search* on the penalty parameter C[5] and the RBF kernel parameter γ, using cross-validation. More rigorously, for the linear and polynomial kernel we selected C from $[10^{-1}, 1, 10, 10^2]$ by using 5-fold cross-validation on the training set. For the RBF kernel we selected C and γ from $[10^{-1}, 1, 10, 10^2]$ and $[10^{-3}, 10^{-2}, 10^{-1}, 1, 10]$, respectively, by using 5-fold cross-validation on the training set.

The accuracy of each classifier is based on the percentage of successful predictions on the test sets of each dataset. As suggested by Friedman et al. (1997), accuracy was measured via the *holdout method* for larger training sets, and via *stratified 5-fold cross-validation* for smaller ones, using the methods described by Kohavi (1995). Throughout the experiments, we used exactly the same folds, hence, the same information was available for training and testing all classifiers. To achieve this, data was previously discretized and shuffled and all evaluated implementations were updated, including the ELR implementation, in order to construct exactly the same folds for a given dataset. Results are presented in Table 4.5 and Table 4.6, where the accuracy is annotated with the standard deviation. The standard deviation is computed according to the binomial formula $\sqrt{acc \times (1 - acc)/m}$, where acc is the classifier accuracy and, for the cross-validation tests, m is the size of the dataset. For the case of holdout tests, m is the size of the test set. In addition, scatter plots of the accuracies of the proposed methods against the others are depicted in Figure 4.4 and Figure 4.5. Points above the diagonal line represent cases where the method shown in the vertical axis performs better than the one on the horizontal axis. Crosses over the points depict the standard deviation.

We compared the performance of the classifiers using Wilcoxon signed-rank tests, using the same procedure as Grossman and Domingos (2004). This test is applicable when paired classification accuracy differences, along the datasets, are independent and non-normally distributed. Alternatively, a paired t-test could be used. However, given that some accuracies are obtained with cross-validation, there is no reason to assume normally distributed classification accuracy differences as the central limit theorem cannot be applied. Furthermore, Wilcoxon

[5]The penalty parameter C required by SVM models controls the trade off between allowing training errors and forcing rigid margins, providing a *soft margin* that allows some misclassifications for non-separable cases.

	Classifier	GHC2	GHC2	TAN	TAN	TAN	TAN
	Struct. Learning	LL	f̂CLL	LL	LL	âCLL	f̂CLL
	Param. Learning	OFE	OFE	OFE	ELR	OFE	OFE
1	australian	85.22 ±1.35	85.51 ±1.34	84.93 ±1.36	84.35 ±1.38	85.51 ±1.34	85.36 ±1.35
2	breast	96.19 ±0.73	97.36 ±0.61	96.19 ±0.73	96.19 ±0.73	97.66 ±0.58	97.66 ±0.58
3	chess	91.72 ±0.84	92.92 ±0.79	92.36 ±0.81	97.09 ±0.51	91.84 ±0.84	93.01 ±0.78
4	cleve	81.42 ±2.26	82.77 ±2.19	81.76 ±2.24	80.79 ±2.29	84.12 ±2.12	82.77 ±2.19
5	corral	98.44 ±1.10	99.22 ±0.78	100.00 ±0.00	100.00 ±0.00	99.22 ±0.78	100.00 ±0.00
6	crx	84.99 ±1.40	86.06 ±1.36	85.45 ±1.38	85.44 ±1.38	86.22 ±1.35	87.14 ±1.31
7	diabetes	78.91 ±1.47	79.17 ±1.47	79.04 ±1.47	78.77 ±1.48	78.12 ±1.49	78.91 ±1.47
8	flare	82.74 ±1.16	82.93 ±1.15	82.55 ±1.16	81.71 ±1.18	80.3 ±1.22	82.55 ±1.16
9	german	73.30 ±1.4	73.90 ±1.39	73.30 ±1.4	73.90 ±1.39	75.80 ±1.35	74.20 ±1.38
10	glass	77.10 ±2.87	78.97 ±2.79	76.64 ±2.89	75.27 ±2.95	73.83 ±3.00	78.97 ±2.79
11	glass2	85.89 ±2.73	85.89 ±2.73	85.89 ±2.73	86.46 ±2.68	85.28 ±2.78	85.89 ±2.73
12	heart	82.59 ±2.31	83.70 ±2.25	81.85 ±2.35	82.22 ±2.33	85.93 ±2.12	83.70 ±2.25
13	hepatitis	86.25 ±3.85	88.75 ±3.53	86.25 ±3.85	88.75 ±3.53	85.00 ±3.99	90.00 ±3.35
14	iris	93.33 ±2.04	94.67 ±1.83	93.33 ±2.04	93.33 ±2.04	94.00 ±1.94	94.00 ±1.94
15	letter	86.14 ±0.49	86.44 ±0.48	86.06 ±0.49	88.96 ±0.44	86.14 ±0.49	86.40 ±0.48
16	lymphography	81.76 ±3.17	85.14 ±2.92	83.11 ±3.08	86.46 ±2.81	83.78 ±3.03	83.11 ±3.08
17	mofn	90.61 ±1.68	90.61 ±1.68	90.90 ±1.66	100.00 ±0.00	90.04 ±1.73	90.90 ±1.66
18	pima	78.26 ±1.49	78.39 ±1.49	78.52 ±1.48	77.74 ±1.50	78.39 ±1.49	78.52 ±1.48
19	satimage	88.54 ±0.71	88.25 ±0.72	87.86 ±0.73	87.60 ±0.74	88.20 ±0.72	88.20 ±0.72
20	segment	95.29 ±0.76	92.49 ±0.95	95.29 ±0.76	95.58 ±0.74	91.17 ±1.02	92.24 ±0.96
21	shuttle	99.85 ±0.09	100.00 ±0.00	99.85 ±0.09	99.84 ±0.09	100.00 ±0.00	100.00 ±0.00
22	soybean	93.42 ±1.05	93.42 ±1.05	92.35 ±1.12	93.24 ±1.06	91.99 ±1.14	93.42 ±1.05
23	vehicle	73.17 ±1.52	72.10 ±1.54	72.58 ±1.53	72.93 ±1.53	70.33 ±1.57	72.10 ±1.54
24	vote	94.48 ±1.09	91.03 ±1.37	94.25 ±1.12	94.94 ±1.05	93.33 ±1.20	91.49 ±1.34
25	waveform	75.28 ±0.63	78.19 ±0.60	75.3 ±0.63	75.34 ±0.63	78.26 ±0.60	77.72 ±0.61

Table 4.5: Accuracy of Bayesian network-based classifiers annotated with the standard deviation.

	Classifier	C4.5	1-NN	3-NN	5-NN	SVM	SVM2	SVMG	LogR
1	australian	85.94 ±1.32	82.46 ±1.45	85.36 ±1.35	85.94 ±1.32	84.78 ±1.37	75.80 ±1.63	82.61 ±1.44	83.62 ±1.41
2	breast	95.90 ±0.76	97.07 ±0.65	96.93 ±0.66	96.93 ±0.66	97.51 ±0.60	96.05 ±0.75	96.63 ±0.69	96.63 ±0.69
3	chess	99.45 ±0.23	94.85 ±0.68	95.22 ±0.65	94.20 ±0.72	96.87 ±0.53	99.26 ±0.26	99.17 ±0.28	97.24 ±0.50
4	cleve	76.69 ±2.46	78.38 ±2.39	80.41 ±2.31	82.77 ±2.19	82.09 ±2.23	72.97 ±2.58	78.38 ±2.39	81.42 ±2.26
5	corral	92.19 ±2.37	92.19 ±2.37	92.19 ±2.37	92.19 ±2.37	89.06 ±2.76	100.00 ±0.00	100.00 ±0.00	88.28 ±2.84
6	crx	85.91 ±1.36	82.70 ±1.48	85.15 ±1.39	86.22 ±1.35	86.98 ±1.32	79.94 ±1.57	82.54 ±1.49	86.37 ±1.34
7	diabetes	77.6 ±1.50	78.12 ±1.49	77.86 ±1.50	77.73 ±1.50	77.47 ±1.51	76.56 ±1.53	77.86 ±1.50	78.65 ±1.48
8	flare	82.27 ±1.17	80.11 ±1.22	81.24 ±1.20	82.65 ±1.16	82.46 ±1.16	82.27 ±1.17	80.49 ±1.21	82.55 ±1.16
9	german	73.00 ±1.40	69.80 ±1.45	70.40 ±1.44	73.20 ±1.40	75.60 ±1.36	66.60 ±1.49	71.40 ±1.43	75.80 ±1.35
10	glass	75.70 ±2.93	79.44 ±2.76	77.10 ±2.87	73.83 ±3.00	75.70 ±2.93	77.10 ±2.87	78.04 ±2.83	73.83 ±3.00
11	glass2	82.82 ±2.95	86.50 ±2.68	83.44 ±2.91	80.37 ±3.11	86.50 ±2.68	87.73 ±2.57	88.34 ±2.51	86.50 ±2.68
12	heart	82.96 ±2.29	83.33 ±2.27	82.59 ±2.31	83.70 ±2.25	84.81 ±2.18	78.52 ±2.50	83.70 ±2.25	84.81 ±2.18
13	hepatitis	85.00 ±3.99	87.50 ±3.70	91.25 ±3.16	92.50 ±2.94	83.75 ±4.12	87.50 ±3.70	87.50 ±3.70	78.75 ±4.57
14	iris	93.33 ±2.04	94.00 ±1.94	94.67 ±1.83	94.67 ±1.83	94.00 ±1.94	92.67 ±2.13	92.67 ±2.13	92.67 ±2.13
15	letter	77.50 ±0.59	90.92 ±0.41	89.60 ±0.43	89.04 ±0.44	89.00 ±0.44	94.20 ±0.33	94.16 ±0.33	86.10 ±0.49
16	lymphography	78.38 ±3.38	83.11 ±3.08	83.11 ±3.08	81.76 ±3.17	82.43 ±3.13	81.76 ±3.17	82.43 ±3.13	69.59 ±3.78
17	mofn	85.58 ±2.03	89.06 ±1.80	86.35 ±1.98	85.48 ±2.03	100.00 ±0.00	99.90 ±0.18	100.00 ±0.00	100.00 ±0.00
18	pima	77.21 ±1.51	76.95 ±1.52	76.82 ±1.52	76.69 ±1.53	78.91 ±1.47	76.95 ±1.52	77.08 ±1.52	78.26 ±1.49
19	satimage	82.33 ±0.85	87.86 ±0.73	87.96 ±0.73	87.82 ±0.73	85.19 ±0.79	88.69 ±0.71	88.25 ±0.72	83.54 ±0.83
20	segment	94.15 ±0.85	94.02 ±0.85	93.38 ±0.9	91.48 ±1.01	94.66 ±0.81	97.33 ±0.58	97.46 ±0.57	94.53 ±0.82
21	shuttle	99.70 ±0.13	99.9 ±0.07	99.75 ±0.11	99.64 ±0.14	99.95 ±0.05	100.00 ±0.00	100.00 ±0.00	99.95 ±0.05
22	soybean	91.28 ±1.19	90.21 ±1.25	89.86 ±1.27	89.32 ±1.30	91.46 ±1.18	91.46 ±1.18	91.99 ±1.14	89.15 ±1.31
23	vehicle	67.73 ±1.61	71.04 ±1.56	71.16 ±1.56	71.39 ±1.55	71.75 ±1.54	74.00 ±1.51	64.54 ±1.64	70.80 ±1.56
24	vote	95.17 ±1.03	92.87 ±1.23	93.56 ±1.18	93.33 ±1.20	93.33 ±1.20	94.02 ±1.14	95.17 ±1.03	92.64 ±1.25
25	waveform	65.49 ±0.69	70.79 ±0.66	73.19 ±0.65	74.68 ±0.63	77.66 ±0.61	80.51 ±0.58	81.89 ±0.56	71.36 ±0.66

Table 4.6: Accuracy of other state-of-the-art classifiers annotated with the standard deviation.

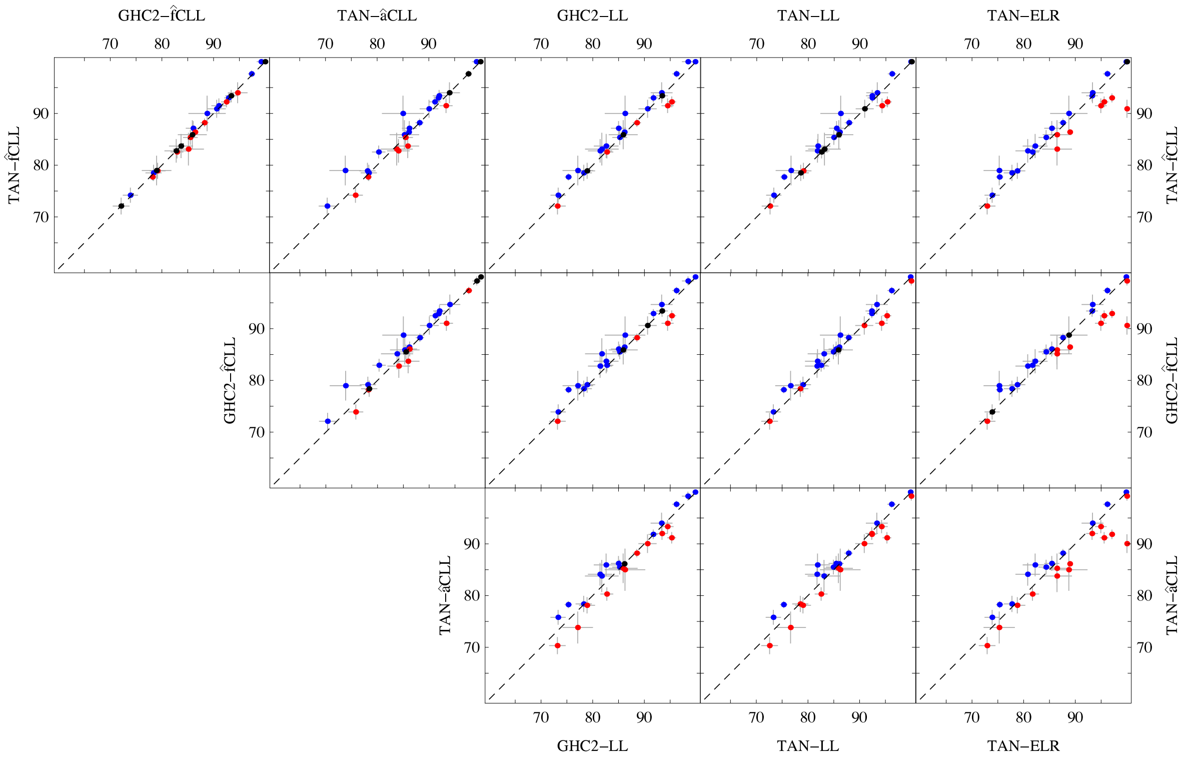


Figure 4.4: Scatter plots of the accuracy of Bayesian network-based classifiers.

signed-rank tests are more conservative than paired t-tests, that is, Wilcoxon signed-rank tests yield non-statistical significance in cases where paired t-tests do. Results are depicted in Table 4.7 and Table 4.8. Each entry of Table 4.7 and Table 4.8 gives the Z-test and p-value of the significance test for the corresponding pairs of classifiers. The arrow points to the superior learning algorithm, in terms of classification rate. A double arrow is used if the difference is significant with p-value smaller than 0.05.

Over all, TAN-f̂CLL-OFE and GHC-f̂CLL-OFE performed the best (Tables 4.7–4.8). They outperformed C4.5, the nearest neighbor classifiers, and logistic regression, as well as the generatively-trained counterparts, TAN-LL-OFE and GHC-LL-OFE, all differences being statistically significant at the $p < 0.05$ level. On the other hand, TAN-âCLL-OFE did not stand out compared to most of the other methods. Moreover, TAN-f̂CLL-OFE and GHC-f̂CLL-OFE classifiers fared sightly better than TAN-LL-ELR and the SVM classifiers, although the difference was not statistically significant. In these cases, the only practically relevant factor is computational efficiency.

It is worthwhile noticing that the GHC2 implementation from WEKA is not restricted to

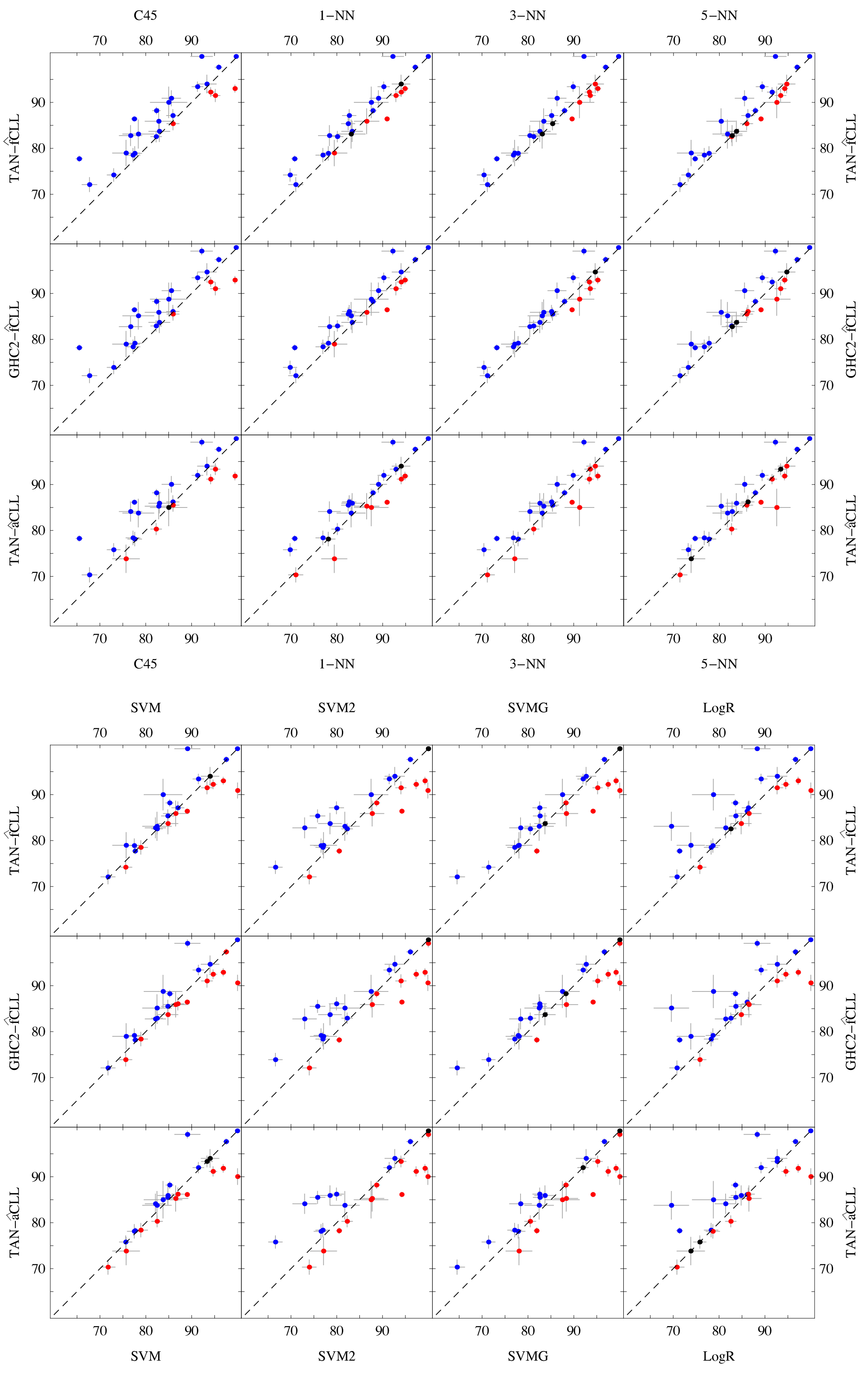


Figure 4.5: Scatter plot of the accuracy of proposed methods against state-of-the-art classifiers.

Classifier	GHC2	TAN	GHC2	TAN	TAN
Struct.	$\hat{\text{f}}$CLL	âCLL	LL	LL	LL
Param.	OFE	OFE	OFE	OFE	ELR
TAN	0.37	1.44	2.13	2.13	0.31
$\hat{\text{f}}$CLL	0.36	0.07	0.02	0.02	0.38
OFE	←	←	⇐	⇐	←
GHC2		1.49	2.26	2.21	0.06
$\hat{\text{f}}$CLL		0.07	0.01	0.01	0.48
OFE		←	⇐	⇐	←
TAN			0.04	-0.34	-1.31
âCLL			0.48	0.37	0.10
OFE			←	↑	↑

Table 4.7: Statistical significance of the results achieved by the Bayesian network-based classifiers according to the Wilcoxon signed-rank test.

Classifier	C4.5	1-NN	3-NN	5-NN	SVM	SVM2	SVMG	LogR
TAN	3.00	2.25	2.16	2.07	0.43	0.61	0.21	1.80
$\hat{\text{f}}$CLL	<0.01	0.01	0.02	0.02	0.33	0.27	0.42	0.04
OFE	⇐	⇐	⇐	⇐	←	←	←	⇐
GHC2	3.00	2.35	2.20	2.19	0.39	0.74	0.11	1.65
$\hat{\text{f}}$CLL	<0.01	<0.01	0.01	0.01	0.35	0.23	0.45	0.05
OFE	⇐	⇐	⇐	⇐	←	←	←	⇐
TAN	2.26	1.34	1.17	1.31	-0.40	-0.29	-0.55	1.37
âCLL	0.01	0.09	0.12	0.09	0.35	0.38	0.29	0.09
OFE	⇐	←	←	←	↑	↑	↑	←

Table 4.8: Statistical significance of the results achieved by the other state-of-the-art classifiers according to the Wilcoxon signed-rank test.

augmented naive Bayes network structures. Therefore, running GHC2 with $\hat{\text{f}}$CLL accounts for empirically verifying the quality of $\hat{\text{f}}$CLL without augmented naive Bayes restrictions. Actually, although the theoretical derivation of the score was intended for augmented naive Bayes classifiers, the class variable does not need to be a parent of the node to compute the score. This is clear from the criterion used to choose the parents for a certain node described in the information-theoretical account of $\hat{\text{f}}$CLL in Section 4.3.2. As a matter of fact, in our experiments, GHC2-$\hat{\text{f}}$CLL-OFE performed significantly better than TAN-LL-OFE, GHC2-LL-OFE, C4.5, *k*-NN and LogR classifiers. Moreover, it also outperforms, although the difference is not statistically significant, SVM's and TAN-âCLL-OLE classifiers. TAN-$\hat{\text{f}}$CLL-OFE was the only classifier that showed to performed better that GHC2-$\hat{\text{f}}$CLL-OFE, but the difference was not statistical significant.

To roughly characterize the computational complexity of learning the various classifiers, we measured the total time required by each classifier to process all the 25 datasets. Reporting the total time instead of the individual times for each dataset will emphasize the significance of the larger data sets. However, the individual times were in accordance with the general conclusion drawn from the total time. Most of the methods only took a few seconds ($\sim 1-3$ seconds), except for TAN-âCLL-OFE which took a few minutes ($\sim 2-3$ minutes), SVM with linear kernel which took some minutes ($\sim 17-18$ minutes), TAN-LL-ELR and SVM with polynomial kernel which took a few hours ($\sim 1-2$ hours) and, finally, LogR and SVM with RBF kernel which took several hours ($\sim 18-32$ hours).

In the case of TAN-âCLL-OFE, the Mathematica package was used. Mathematica is a symbolic language being per se computationally inefficient. Nevertheless, in theory, TAN-âCLL-OFE classifiers should have the same computational cost as TAN-LL-OFE, or TAN-$\hat{\text{f}}$CLL-OFE, being both algorithms quadratic in the number of features and linear in the number of instances. We attribute the computational cost of TAN-âCLL-OFE to the implementation on Mathematica, rather than to the method. In what concerns TAN-LL-ELR, the ELR discriminative parameter learning is computationally more expensive than $\hat{\text{f}}$CLL-based discriminative learning. In our experiments, TAN-LL-ELR was 3 order of magnitude slower than TAN-$\hat{\text{f}}$CLL-OFE. Su and Zhang (2006) had already reported a difference of 6 orders of magnitude, but different datasets were used in their experiments.

We also verified that selection of model parameters in SVMs is extremely time demanding and, in our 25 experiments, the linear kernel was 1 order of magnitude faster than the polynomial kernel and 2 orders of magnitude faster than the RBF kernel. However, the linear kernel is already 2 orders of magnitude slower than $\hat{\text{f}}$CLL-based classifiers. The LogR and SVM with RBF kernel classifiers were the most time demanding ones, being 4 orders of magnitude slower than $\hat{\text{f}}$CLL-based classifiers. Furthermore, in terms of memory, SVM's with polynomial and RBF kernels, as well as LogR, required 1GB of memory, whereas all other classifiers cope with the standard 128MB. With this empirical analysis, we conclude that $\hat{\text{f}}$CLL scoring criterion was able to produce effective classifiers, at least comparable with state-of-the-art ones as ELR, SVM's and LogR, but did so with a factor of $\sim 500-52000$ speedup with minimum memory requirements.

Chapter 5

CκG: Learning consistent κ-graphs

The contribution of this section consists in a new probabilistic model, based on Bayesian networks, for TFBS representation which takes into account dependencies among binding sites. We start by discussing in Section 5.1 the biological motivation for the usage of Bayesian networks in this context. Next, in Section 5.2, we present some trends and issues related to the efficient learning of Bayesian networks and propose, in Section 5.3, a new class of Bayesian networks, exponentially larger than trees, which can still be learned in polynomial-time. We called this class *consistent κ-graph* (CκG) Bayesian networks. In Section 5.3.1, we induce a classifier from the CκG Bayesian network learning algorithm capable of classifying TFBS's from a collection of aligned sequences. In the subsequent Section 5.3.2, we provide a full comparison of the expressiveness of the different Bayesian network models employed to describe TFBS. We conclude this chapter by presenting in Section 5.3.3 an algorithm for discriminative learning of two-component mixtures of CκG Bayesian networks, arising from this effort a new scoring criterion called *mixture-based factorized conditional log-likelihood* (m$\hat{\text{f}}$CLL). Finally, we present experimental results in Section 5.4.

5.1 Biological motivation for CκG Bayesian networks

Probabilistic models of TFBS described in Section 2.2.2 can be represented as Bayesian network classifiers. The simplest one is the PSSM that, as noticed by Barash et al. (2003), together with a background model corresponds to a Naive Bayes classifier with a binary class variable. The attributes of the classifier are the nucleotide positions within the motif, and given that the class variable takes value 1, the distribution over the attributes is given by the PSSM. The background model is usually a uniform distribution, that is, given that the class variable takes value 0, the distribution over the attributes is uniform.

The Markov chain models used to overcome the strong independence assumptions of the PSSM model are also Bayesian network classifiers where the topology of the network is restricted in some way. In the case first-order Markov chains, a total order between the positions is assumed (the adjacency order), and one position depends only of the previous one. For the case of n-th Markov chains, dependencies in the previous n positions are allowed whereas for the case of VLMM's the number of dependencies at each position may vary. Finally, PVLMM's just assume that the total order does not need to be the adjacency order. We can conclude that finding probabilistic models of TFBS is closely connected to learning Bayesian networks. This relationship was acknowledged by Barash et al. (2003), where TFBS models are based on TAN classifiers.

A very important (and mandatory) feature that probabilistic models of TFBS must fulfill is that learning them must be efficient. Therefore, taking into account the above discussion, it seems that the best probabilistic TFBS model is the largest set of Bayesian networks that can be learned in polynomial-time. Unfortunately, the results concerning efficient learning of Bayesian networks are very restrictive, and tree structures seem to be the upper bound of efficient Bayesian network learning. In the following, we discuss such hardness results and then propose a new class of Bayesian networks, exponentially larger than trees, that can still be learned in polynomial-time. We called this class CκG Bayesian networks and we will use it afterwards to model TFBS.

5.2 Issues and trends in efficient learning of Bayesian networks

Learning unrestricted Bayesian networks from data under typical scoring criteria is NP-hard (Chickering et al., 2004). Consequently, the standard methodology for addressing the problem of learning Bayesian networks became heuristic search, based on scoring metrics optimization, conducted over some search space. Many algorithms have been proposed along these lines, varying both on the formulation of the search space (network structures, equivalence classes of network structures and orderings over the network variables), and on the algorithm to search the space (greedy hill-climbing, simulated annealing, genetic algorithms, tabu search, etc). Although searching in the space of network structures was commonly considered as the standard choice, more recently it has been shown that searching the space of orderings empirically outperforms the standard baseline of greedy hill-climbing, modified with a tabu list and random restarts, over the space of network structures (Teyssier and Koller, 2005).

There are several reasons why orderings have recently been attracting so much attention (Teyssier and Koller, 2005; Friedman and Koller, 2003). First, orderings provide a first clue on the causality of the network variables, which can then be refined in subsequent processing. By itself, this observation is of limited use, since determining an appropriate ordering is a difficult problem. Nevertheless, if the search is conducted in the space of orderings, instead of networks structures, there is a severe decrease in the search space, which simplifies the task. Second, given an ordering on the network variables, finding an optimal bounded in-degree network consistent with it is not NP-hard. Indeed, if the in-degree of a node is bounded to κ, this task can be accomplished in $O(n^\kappa)$ time, where n is the number of variables in the network. Finally, given a network consistent with some ordering on the variables, there is no need to check for network cycles, since it is guaranteed that the network will always be acyclic.

The contribution of the following sections consists in taking a topological sorting of the optimal tree Bayesian network as a heuristic for a causality order between the network variables. An efficient candidate for the required topological sorting is the total order induced by the breadth-first search over the optimal tree, although, any topological sorting can be used. The main idea is to take a topological sorting over an optimal tree Bayesian network and then search for an optimal bounded in-degree network consistent with it. The search space of this procedure is a subclass of Bayesian networks which is more general than trees and intersect, but is not contained in, polytrees (recall that a polytree is a DAG in which there are not two different paths from one node to another). This search space consists of directed acyclic graphs of in-degree at most κ that are consistent with the chosen topological sorting of the optimal tree Bayesian network, henceforward called *consistent κ-graphs* (CκG).

The foremost benefit of this approach is that learning CκG Bayesian networks can be done efficiently, that is, in polynomial time over the number of network variables. Moreover, the class of networks consistent with a topological sorting is exponentially larger, in the number of variables, when compared to tree Bayesian networks. The proposed algorithm copes with scoring functions that decompose over the network structure, as the $\hat{\text{f}}$CLL score presented in the previous chapter. We show that the score of the resulting network is always greater than or equal to the score of an optimal tree Bayesian network.

5.3 Consistent Bayesian networks

A natural approach to efficiently learn structures more complex than trees is to compute the optimal tree Bayesian network and then allow dependencies consistent with the topological order induced by this tree. Since, in particular, the optimal tree Bayesian network is consistent with its topological order, the resulting optimal Bayesian network will score always better than or the same as the optimal tree. For presenting these results we need to introduce some auxiliary concepts and notation (refer to Table 5.1).

Symbol	**Meaning**
T	training data, $T = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$
N	number of instances in T
$\mathbf{X}$	set of attributes/nodes, $\mathbf{X} = \{X_1, \ldots, X_n\}$
n	number of attributes/nodes
κ	in-degree of the considered network graphs
$\prec$	parenthood relation
$\sqsubseteq$	topological sorting
R	tree Bayesian network
$\mathcal{B}_R^\kappa$	set of all CκG's w.r.t. the canonical topological sorting $(\mathbf{X}, \sqsubseteq)$ of R
ϕ_i	contribution of X_i to the decomposable score ϕ
Π_{X_i}	parents of X_i in the considered network graphs
C	class variable

Table 5.1: Definition of terms used in Chapter 5.

A *κ-graph* is a graph where each node has in-degree at most κ. Trees and forests are 1-graphs. A tree induces a partial order by taking the reflexive and transitive closure of the parenthood relation $\prec$ where $X_i \prec X_j$ if and only if X_i is parent of X_j. We call this partial order the *topological order* of the tree. A *topological sorting $\sqsubseteq$ of a tree Bayesian network R* is a linear (also known as total) ordering $\sqsubseteq$ between the nodes in R such that it contains the topological order of R, that is, if $X_i \prec X_j$ then $X_i \sqsubseteq X_j$. In general, a tree may induce several topological sortings since the topological order may be linearized in several different ways. Any of these possible linear orders (linearized from the induced topological order) are considered a topological sorting of R. The order that nodes are visited in a *breadth-first search* (BFS) of R induces a possible topological sorting. An acyclic graph $G = (\mathbf{X}, E)$ is said to be *consistent with a topological order* $(\mathbf{X}, \sqsubseteq)$ if $X_i \sqsubseteq X_j$ whenever there is an edge from X_i to X_j in G.

Definition 5.3.1 (Consistent κ-graph) Given a tree Bayesian network R with a set of attributes N and a topological sorting $(\mathbf{X}, \sqsubseteq)$ of R, a graph $G = (\mathbf{X}, E)$ is said to be a *consistent κ-graph* (CκG) w.r.t. $(\mathbf{X}, \sqsubseteq)$ if it is a κ-graph and for any edge in E from X_i to X_j we have that $X_i \sqsubseteq X_j$.

From this point on we assume that from any tree we have a canonical way to induce its topological sorting. As already pointed out, a linear way to do so is by considering the BFS order. Assuming the BFS order the definition of consistency imposes that there can only exist an edge from X_i to X_j in a consistent κ-graph G if X_i is less than or as deep as X_j in R. For the sake of presentation, we assume that if $i < j$ and X_i and X_j are at the same level, then the BFS over R reaches X_i before X_j. However, there are better ways of ordering nodes at the same level. A simple approach is to consider a random order. A better one is to compute the optimal branching solely for tree nodes at the same level and order them with a BFS over the resulting branching (eventually, applying this procedure recursively). An alternative approach is to use heuristic methods based on inter-position dependence, as measured by χ^2 values, and choose an order that maximizes this dependency. We do not discuss in detail which is the best canonical topological sorting since, in practice, we verified that the results were insensitive to that choice. Nevertheless, we assume that the topological sorting can be computed in linear time. Henceforward, we denote by $\mathcal{B}_R^\kappa$ the set of all CκG's w.r.t. the canonical topological $(\mathbf{X}, \sqsubseteq)$ sorting of R.

Proposition 5.3.2 (CκG acyclicity) Any consistent κ-graph w.r.t. any topological sorting of a tree Bayesian network is acyclic.

Proof: By absurd, assume that the consistent κ-graph has a non-trivial cycle. This means that $X_i \sqsubseteq X_j$ for all X_i and X_j in the cycle which contradicts the linearity of the topological sorting of the tree. □

Example 5.3.3 Given the underlying graph for the attributes of a tree Bayesian network R in Figure 5.1(a), its BFS order is represented by a dashed line in Figure 5.1(b). Three C2G w.r.t. the BFS order of R are presented in Figure 5.1(c), (d) and (e). Observe that the graph in Figure 5.1(c) is not a polytree. Indeed, there are two different paths from node 1 to 4: $1 \to 2 \to 4$ and $1 \to 3 \to 4$. On the other hand, the graph in Figure 5.1(d) is a C2G that is also a polytree. Finally, both naive Bayes in Figure 5.1(e) and the tree Bayesian network in Figure 5.1(a) are C2G w.r.t. the BFS order of R.

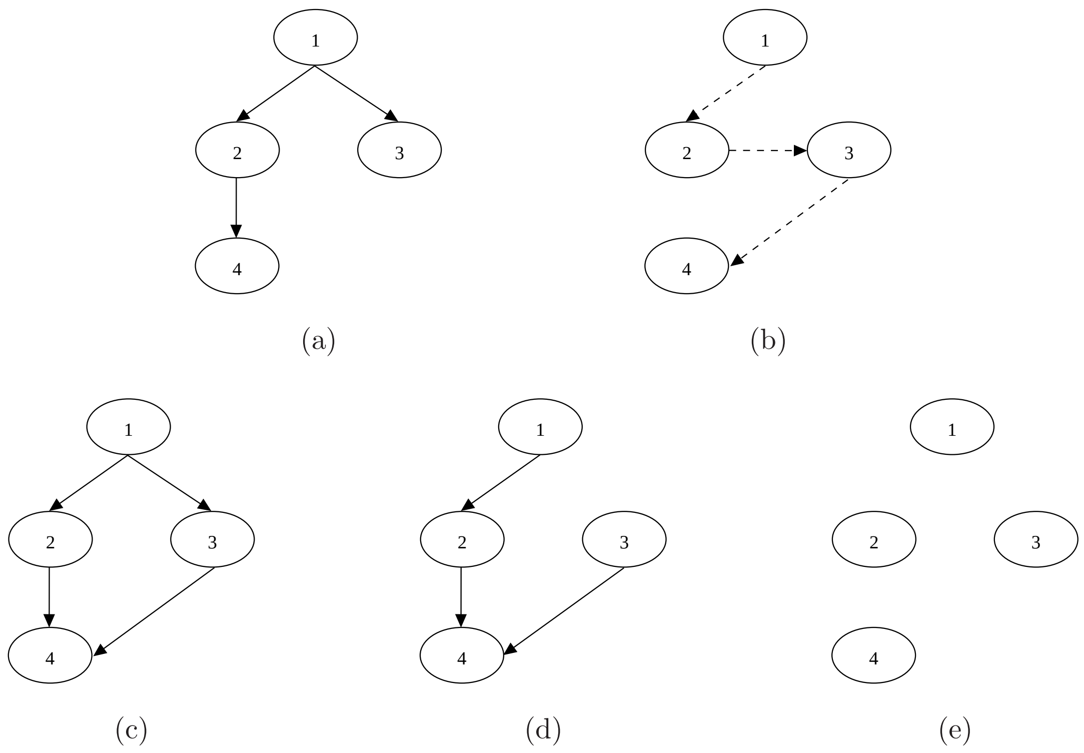


Figure 5.1: Figure relative to the Example 5.3.3.

The core idea of the CκG learning algorithm is to compute an optimal tree Bayesian network R and improve it by adding/removing dependencies which were omitted/present because of the tree structure restrictions. For efficiency purposes, the modified model must be a consistent κ-graph w.r.t. the canonical topological sorting of R. In this context, the canonical topological sorting of R might add dependencies from higher nodes to deeper nodes.

In detail, the algorithm starts by computing an optimal tree Bayesian network (Algorithm 3.1, 3.2 or 3.3 depending on the score being considered) as described in Section 3.1.1. Then it computes the canonical topological sorting over the optimal tree to construct a linear order. Finally, it ranges over each variable X_i, generates the set α_i of all variables less than X_i, and takes as parents of X_i the set $S \subseteq \alpha_i$ such that $\phi_i(S,T)$ is maximal over all subsets of α_i with at most κ variables. The pseudocode of the algorithm is presented in Algorithm 5.1.

Algorithm 5.1 Learning CκG BN's, decomposable ϕ-score

```
1. run a (deterministic) algorithm that outputs an optimal tree Bayesian network $R$ according to $\phi$
2. compute the canonical topological sorting $(\mathbf{X}, \sqsubseteq)$ of $R$
3. for each variable $X_i$ in $R$ do
    (a) compute the set $\alpha_i = \{X_j \in R : X_i \sqsubseteq X_j \text{ and } X_i \neq X_j\}$
    (b) for each subset $S$ of $\alpha_i$ with at most $\kappa$ variables do
        i. compute $\phi_i(S,T)$
        ii. if $\phi_i(S,T)$ is the maximal score for $X_i$ then $\Pi_{X_i} := S$
4. return the directed graph $G$ such that the parents of a variable $X_i$ are $\Pi_{X_i}$
```

Theorem 5.3.4 (Soundness) Algorithm 5.1 constructs a CκG Bayesian network that maximizes the ϕ-score given data T.

Proof: Since all potential parents for each node are checked, the algorithm returns the κ-graph consistent w.r.t. the canonical topological sorting $(\mathbf{X}, \sqsubseteq)$ of R with the highest score. Moreover, α_i is defined in such a way that potential parents for X_i are those which are smaller than X_i, according to $(\mathbf{X}, \sqsubseteq)$, thus, by Proposition 5.3.2 the resulting graph is acyclic. □

Proposition 5.3.5 (Optimality) Algorithm 5.1 constructs a CκG Bayesian network whose ϕ-score is always greater than, or equal to, the ϕ-score of the optimal tree Bayesian network.

Proof: Start by noticing that the soundness of Algorithm 5.1 assures that the resulting CκG w.r.t. the canonical topological sorting of R, say B, is maximal among all CκG's in $\mathcal{B}_R^\kappa$. Moreover, observe that the underlying graph of the tree Bayesian network R is consistent w.r.t. the canonical topological sorting of R, that is, $R \in \mathcal{B}_R^\kappa$ for all $k \geq 1$. Hence, the soundness of Algorithm 5.1 guarantees that $\phi(B,T) \geq \phi(R,T)$. □

Theorem 5.3.6 (Complexity) Algorithm 5.1 constructs an optimal CκG Bayesian network in $O(n^{\kappa+1}\gamma(\kappa, T))$ time where $\gamma(\kappa, T)$ is an upper bound for computing $\phi_i(S, T)$.

Proof: Step 2 takes $O(n)$ time. Step 3a) takes $O(n)$ time, while Step 3b) takes $O(n^{\kappa}\gamma(S, T))$ time because it ranges over all subsets S with at most κ elements (which takes $O(n^{\kappa})$ time) and for each of this sets it computes $\phi_i(S, T)$ (which takes $O(\gamma(S, T)$ time). Thus, the overall time complexity of the algorithm is $O(n^{\kappa+1}\gamma(S, T))$. $\square$

The theorems above assert the soundness and polynomial-time bound of the CκG learning algorithm. At this point it remains to show that, despite considering an optimal tree to confine the search space, the number of graphs searched increases exponentially, in the number of variables, when compared to trees.

Proposition 5.3.7 (Expressiveness) Let R be a tree with n variables, then the number of non-trees in $\mathcal{B}_R^{\kappa}$ is at least $2^{n\kappa-\frac{\kappa^2}{2}-\frac{\kappa}{2}-1}$ when $n \geq \kappa$.

Proof: We denote by $(\mathbf{X}, \sqsubseteq)$ the canonical topological sorting of R. Since, this order is total, for any pair of nodes X_i and X_j in $\mathbf{X}$, with $i \neq j$, we can say that a node X_i is *lower than* X_j if and only if $X_i \sqsubseteq X_j$. Given this, notice that the i-th node of R has precisely $(i-1)$ lower nodes. We conclude that, when $i > \kappa(\leq n)$, there are at least 2^{κ} subsets of $\mathbf{X}$ with at most κ lower nodes. Moreover, when $(1 \leq)i \leq \kappa$, only 2^{i-1} subsets of $\mathbf{X}$ with at most κ lower nodes exist. Thus,

$$|\mathcal{B}_R^{\kappa}| \geq \left(\prod_{i=\kappa+1}^{n} 2^{\kappa}\right) \times \left(\prod_{i=1}^{\kappa} 2^{i-1}\right) = 2^{n\kappa-\frac{\kappa^2}{2}-\frac{\kappa}{2}}$$

give us a lower bound for the total number of possible CκG w.r.t. the canonical topological sorting of R (recall that a C1G is also a C2G, both a C1G and a C2G are also a C3G, and so on). Now, consider that X_i is the root, and X_j is the lowest child of the root in R. The only two subsets of $\mathbf{X}$ with at most κ lower elements than X_j are $\emptyset$ and $\{X_i\}$. This choice splits in two all CκG's in $\mathcal{B}_R^{\kappa}$. Those for which the set of parents of X_j is $\emptyset$ cannot be trees since X_i has no parents as well. Therefore, there are at least $\frac{|\mathcal{B}_R^{\kappa}|}{2} \geq 2^{n\kappa-\frac{\kappa^2}{2}-\frac{\kappa}{2}-1}$ in $\mathcal{B}_R^{\kappa}$ that are non-trees. $\square$

Next we apply the above results to classification towards efficient modeling of TFBS.

5.3.1 CκG classifier: An extension to the TAN classifier

Taking into account the CκG Bayesian network presented in the previous section, and the usage of Bayesian networks in the context of classification, presented in Section 3.1.2, we introduce the CκG Bayesian network classifier. As expected, a CκG Bayesian network classifier is a Bayesian network classifier for which the underlying graph is confined to be consistent with the BFS order of an optimal TAN and to have a bounded in-degree κ.

The algorithm to learn CκG Bayesian network classifiers, for any decomposable score, is presented in Algorithm 5.2. Notice that there are two main differences between the classification algorithm (Algorithm 5.2) and the learning algorithm (Algorithm 5.1). First, in Step 1 it is used one of the algorithms to learn an optimal TAN Bayesian network classifier (Algorithm 3.4, Algorithm 3.5 or Algorithm 3.6, depending on the scoring function being considered). Second, as explained in Section 3.1.2, a Bayesian network classifier has a special attribute C, the class variable, that is parent of all other attributes. Thus, to devise an algorithm to learn CκG Bayesian network classifiers we have to adapt the weights computed in Algorithm 5.1 in such a way that all nodes have C as parent. This is done in Step 3b) where we take as parents of X_i the set $S \subseteq \alpha_i$ such that $\phi_i(S \cup \{C\}, T)$ is maximal over all subsets of α_i with at most κ attributes.

Algorithm 5.2 Learning CκG BNC's, decomposable ϕ-score

```
1. run a (deterministic) algorithm that outputs an optimal TAN Bayesian network classifier $R$ according to $\phi$
2. compute the canonical topological sorting $(\mathbf{X}, \sqsubseteq)$ of $R$ (ignoring the class variable)
3. for each attribute $X_i$ in $R$ do
    (a) compute the set $\alpha_i = \{X_j \in R : X_i \sqsubseteq X_j \text{ and } X_i \neq X_j\}$
    (b) for each subset $S$ of $\alpha_i$ with at most $\kappa$ attributes do
        i. compute $\phi_i(S \cup \{C\}, T)$
        ii. if $\phi_i(S \cup \{C\}, T)$ is the maximal score for $X_i$ then $\Pi_{X_i} := S \cup \{C\}$
4. return the directed graph $G$ such that the parents of an attribute $X_i$ are $\Pi_{X_i}$
```

The soundness and complexity of the algorithm is a simple corollary of Theorems 5.3.4 and 5.3.6, respectively.

Corollary 5.3.8 (Soundness) Algorithm 5.2 constructs a CκG Bayesian network classifier that maximizes the ϕ-score given data T.

Corollary 5.3.9 (Complexity) Algorithm 5.2 constructs a CκG Bayesian network classifier in $O(n^{\kappa+1}\gamma(\kappa, T))$ time where $\gamma(\kappa, T)$ is an upper bound for computing $\phi_i(S \cup \{C\}, T)$.

Moreover, observe that the NB acyclic graph (recall Definition 3.1.12 at page 33) is consistent with any topological sorting. Moreover, since we consider the canonical topological sorting induced by a tree generated by the TAN Bayesian network learning algorithm, the TAN itself is consistent with this sorting. For this reason, the score of a CκG network is always greater than or equal to the score of both NB and TAN networks.

Corollary 5.3.10 (Optimality) Algorithm 5.2 constructs a CκG Bayesian network classifier whose ϕ-score is always greater than, or equal to, the ϕ-score of the optimal TAN and NB Bayesian network classifiers.

5.3.2 Expressiveness of CκG Bayesian networks

Figure 5.2 presents the search space of the different Bayesian network learning algorithms discussed before. Observe that the Algorithm 5.2 starts by finding the optimal TAN, and then computes its canonical topological sorting (ignoring the class variable). When finding the best CκG with respect to the scoring function ϕ, for some data T, the NB will be checked (see Figure 5.1(e)), together with all polytrees that are consistent with the canonical topological sorting (see Figure 5.1(d)), as well as all other consistent Bayesian networks (see Figure 5.1(c)). It is worthwhile noticing that the search space of Algorithm 5.2 differs for each scoring function ϕ and data T. However, this search space will always contain NB, TAN's and all polytrees that are consistent with the canonical topological sorting.

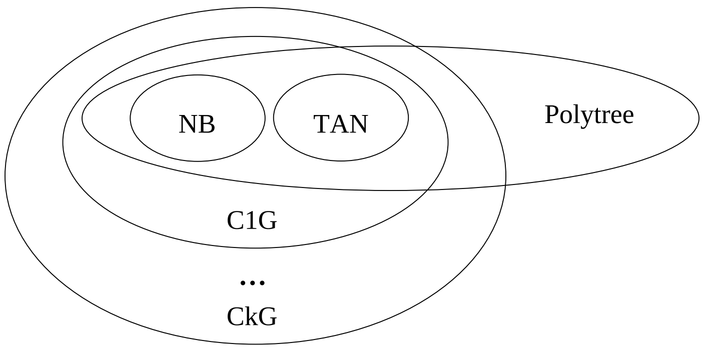


Figure 5.2: Expressiveness of the network models discussed in this work.

Polytrees include both NB and TAN models, and intersect the search space of the algorithm to learn CκG classifiers.

The next result shows that, the search space of Algorithm 5.2 covers Bayesian networks that represent any joint probability distribution when $\kappa = n - 1$.

Theorem 5.3.11 (Completeness of C$(n-1)$G) The search space of Algorithm 5.2 for $\kappa = n-1$ covers Bayesian networks that represent any n-dimensional joint probability distribution.

Proof: Note that, for any topological sorting, there is a consistent $(n-1)$ graph that is complete. Since complete graphs can represent any joint probability distribution, the result follows straightforwardly. $\square$

5.3.3 Discriminative learning of two-component mixtures of CκG Bayesian networks

Herein, we enrich this chapter with mixtures of CκG Bayesian networks since, as we shall see next, mixtures are considerably more natural to use when modeling TFBS's.

Probabilistic mixtures of general graphical models were introduced by Geiger and Heckerman (1996) and since then they have been utterly applied in several domains (Meila and Jordan, 2000; Friedman et al., 1997). Mixtures of arbitrary graphical models are also called *Bayesian multinets*. The main advantage of Bayesian multinets is that they allow to represent *context-specific independences*. We found this context-specific independences when a subset of variables exhibit certain conditional independences for some, but not all, values of a conditional variable.

For convenience, we introduce a few additional notations that apply to Bayesian multinets intended to be learned from data $T = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, where $\mathbf{y}_t = (y_t^1, \ldots, y_t^n, c_t)$. Start by recalling that the class variable C ranges over a finite set, say $\mathcal{C} = \{z_1, \ldots, z_s\}$. It is useful to associate to each index of an instance of T the corresponding class value. More precisely, consider the map

$$\eta : \{1, \ldots, N\} \to \mathcal{C}, \quad \text{where} \quad \eta(t) = c_t.$$

Moreover, we denote by $\mathbf{y}_t^- = (y_t^1, \ldots, y_t^n)$,

$$I_c = \eta^{-1}(z_c) \quad \text{and} \quad T_c = \{\mathbf{y}_t^- : t \in I_c\},$$

for $c \in \{1, \ldots, s\}$. Loosely speaking, I_c is the set of indexes of the instances in T where the class variable takes the value z_c and T_c is the set of instances, excluding the class variable, with the indexes in I_c. Refer to Table 5.2 to a summary of the terms used in this section.

When Bayesian multinets are used for classification tasks, different Bayesian networks B_c are found for each value z_c of the class variable. The Bayesian network found for each value z_c is called the *local Bayesian network* for z_c. The family of local Bayesian networks, endowed with a *mixing proportion* over the class variable, $\lambda_C = P(C)$, constitutes a Bayesian multinet.

Definition 5.3.12 (s-component Bayesian multinet) A *s-component Bayesian multinet* is a tuple $\mathcal{M} = \langle \{\lambda_c\}_{c=1,\ldots,s}, B_1, \ldots, B_s \rangle$ where $\lambda_c = P(C = z_c)$ and B_c is a Bayesian network over $X_1, \ldots, X_n$ for all $c = 1, \ldots, s$.

Bayesian multinets define a unique joint probability distribution given by:

$$P_{\mathcal{M}}(X_1, \ldots, X_n, C) = \lambda_C P_{B_C}(X_1, \ldots, X_n).$$

The standard procedure to learn Bayesian multinets is to compute from data

$$\hat{\lambda}_c = \hat{P}_T(C = z_c) = \frac{N_c}{N}$$

given by the *observed frequency estimates* (OFE), and then learn each B_c independently over the subset T_c. Predictions are made by choosing the class variable that maximizes the posterior probability $P_{\mathcal{M}}(C \mid X_1, \ldots, X_n)$.

The focus of this thesis is on motif representation. There is no reason to think that a unique CκG model is suitable to represent the promoter regions of co-regulated genes – the *background* – and, at the same time, a motif within such promoter region – the *foreground*. Indeed, there seems to be two separate underlying regimes, so instead we model the background and the foreground with different CκG models. In this case, the resulting model is a *two-component mixture of CκG Bayesian network* models. *Mixtures of tree Bayesian networks* are known to perform well when generatively learned (Barash et al., 2003; Meila and Jordan, 2000; Friedman et al., 1997) so we are expecting that mixtures of CκG Bayesian networks also do.

We now capitalize on the work presented in Chapter 4, and extend it to devise a new scoring criterion to learn mixtures of CκG Bayesian networks for classification tasks. As in the previous chapter, we focus our attention in binary classification, that is, we consider the

Symbol	**Meaning**
$\mathcal{M}$	Bayesian multinet (aka Bayesian network mixture)
λ_c	mixing proportion for the c-th component of $\mathcal{M}$
B_c	c-th component of the Bayesian network mixture of $\mathcal{M}$
$\mathcal{C}$	finite domain of the class variable C, $\mathcal{C} = \{z_1, \ldots, z_s\}$
z_c	c-th value that the class variable C can take
N_c	number of instances in T where $C = z_c$
P_{B_c}	joint distribution of $\mathbf{X}$ induced by B_c
$\hat{P}_T$	joint distribution of $(\mathbf{X}, C)$ induced by the OFE parameters
$\mathbf{y}_t$	t-th instance of T, $\mathbf{y}_t = (y_t^1, \ldots, y_t^n, c_t)$
I_c	set of indexes of the instances in T where $C = z_c$
T_c	set of instances, excluding the class variable, with the indexes in I_c
$\mathbf{y}_t^-$	t-th instance of T excluding the class variable, $\mathbf{y}_t^- = (y_t^1, \ldots, y_t^n)$
$(y_t^1, \ldots, y_t^n, 1 - c_t)$	dual of the t-th instance in T (may not occur in T)
U_t	probability of the t-th instance in T
V_t	probability of the dual of the t-th instance in T
x_{ik}	k-th value that the attribute X_i can take
r_i	number of values X_i can take
$\Pi^1_{X_i}$	parents of X_i in B_1
w^1_{ij}	j-th (parent) configuration of $\Pi^1_{X_i}$
q^1_i	number of possible parent configurations of $\Pi^1_{X_i}$
N^1_{ij1k}	number of instances in T where $X_i = x_{ik}$, $\Pi^1_{X_i} = w^1_{ij}$ and $C = 1$
N^1_{ij0k}	number of instances in T where $X_i = x_{ik}$, $\Pi^1_{X_i} = w^1_{ij}$ and $C = 0$
N^1_{ij1}	number of instances in T where $\Pi^1_{X_i} = w^1_{ij}$
G_1	DAG underlying B_1

Table 5.2: Definition of terms used in Section 5.3.3.

two-component mixture of CκG models given by $\mathcal{M} = \langle \lambda_0, B_0, B_1 \rangle$. We omit the λ_1 from the model $\mathcal{M}$ as $\lambda_1 = 1 - \lambda_0$. Moreover, we only address learning B_1 discriminatively, and assume that both the mixing proportion λ_0 and the background model B_0 are generatively learned. The rationale for this approach is that learning the background can be accomplished generatively, since usually the data for the background is very large. Moreover, learning the foreground should be performed discriminatively since we want to distinguish it from the background and, furthermore, the data is scarce.

It is convenient to extend the notation introduced in the previous chapter to cope with the discriminative learning of B_1 within a mixture $\mathcal{M} = \langle \lambda_0, B_0, B_1 \rangle$. In what follows, the usage of the superscript 1 means that we are considering only the Bayesian network B_1 to determine the dependencies between the attributes. We denote by $\Pi^1_{X_i}$ the parents of X_i in B_1 and by q^1_i the number of parent configurations of $\Pi^1_{X_i}$. Moreover, we denote by N^1_{ij1k} the number of instances in the data T where the variable X_i takes its k-th value, the attributes in $\Pi^1_{X_i}$ take their j-th configuration w^1_{ij} and the class variable C takes the value 1; N^1_{ij0k} is defined accordingly as the number of instances in the data T where the variable X_i takes its k-th value, the attributes in $\Pi^1_{X_i}$ take their j-th configuration w^1_{ij} and the class variable C takes the value 0. Similarly, N^1_{ij1} denotes the number of instances in the data T where the attributes in $\Pi^1_{X_i}$ take their j-th configuration w^1_{ij} and the class variable takes value 1. In this case, the *maximum likelihood* (ML) estimates in Equation (3.4) at page 30 become now

$$\hat{\theta}_{ij1k} = \hat{P}_{T_1}(X_i = x_{ik} \mid \Pi^1_{X_i} = w^1_{ij}) = \frac{N^1_{ij1k}}{N^1_{ij1}}. \tag{5.1}$$

For bi-classification tasks in the context of a Bayesian multinet $\mathcal{M} = \langle \lambda_0, B_0, B_1 \rangle$ we have that

$$P_{\mathcal{M}}(c_t \mid y^1_t, \ldots, y^n_t) \;=\; \frac{\lambda_{c_t} P_{B_{c_t}}(y^1_t, \ldots, y^n_t)}{\lambda_{c_t} P_{B_{c_t}}(y^1_t, \ldots, y^n_t) + \lambda_{(1-c_t)} P_{B_{(1-c_t)}}(y^1_t, \ldots, y^n_t)}. \tag{5.2}$$

To simplify notation, consider that

$$U_t = \lambda_{c_t} P_{B_{c_t}}(y^1_t, \ldots, y^n_t) \qquad \text{and} \qquad V_t = \lambda_{(1-c_t)} P_{B_{(1-c_t)}}(y^1_t, \ldots, y^n_t),$$

hence, expression (5.2) can be rewritten as

$$P_{\mathcal{M}}(c_t \mid y^1_t, \ldots, y^n_t) \;=\; \frac{U_t}{U_t + V_t}.$$

In this case, the conditional log-likelihood of T for $\mathcal{M}$ has the following form:

$$\mathrm{CLL}(\mathcal{M} \mid T) = \sum_{t=1}^{N} \log\left(\frac{U_t}{U_t + V_t}\right).$$

To efficiently discriminate between the foreground and the background we need to derive a decomposable scoring criterion. Unfortunately, $\log(U_t + V_t)$ does not decompose over the mixture components B_0 and B_1, but $\log(U_t)$ and $\log(V_t)$ do. Following the same reasoning as in Section 4.3 (page 62), we propose a *minimum variance unbiased* (MVU) approximation

$$\hat{f}(U_t, V_t) = \alpha \log(U_t) + \beta \log(V_t) + \gamma,$$

of the original function

$$f(U_t, V_t) = \log\left(\frac{U_t}{U_t + V_t}\right),$$

when U_t and V_t are probabilities.

By taking the approximation given by Theorem 4.3.1 (page 65), we have that

$$\mathrm{CLL}(\mathcal{M} \mid T) = \sum_{t=1}^{N} \log\left(\frac{U_t}{U_t + V_t}\right) \approx \sum_{t=1}^{N} \alpha \log(U_t) + \beta \log(V_t) + \gamma \tag{5.3}$$

where constants α, β and γ are given by Equations (4.13), (4.14) and (4.15), respectively.

Assuming that both the mixing proportion λ_0 and the background model B_0 are fixed, we only need to learn the foreground model B_1. In this case, detaching the contribution of each instance in T according to the value of its class variable, the right-hand side of the approximation (5.3) can be simplified to

$$\left(\sum_{t\in I_1} \alpha \log(\lambda_1 P_{B_1}(\mathbf{y}_t^-)) + \beta \log(\lambda_0 P_{B_0}(\mathbf{y}_t^-)) + \gamma\right) + \left(\sum_{t\in I_0} \alpha \log(\lambda_0 P_{B_0}(\mathbf{y}_t^-)) + \beta \log(\lambda_1 P_{B_1}(\mathbf{y}_t^-)) + \gamma\right)$$
$$= \left(\sum_{t\in I_1} \alpha \log(P_{B_1}(\mathbf{y}_t^-))\right) + \left(\sum_{t\in I_0} \beta \log(P_{B_1}(\mathbf{y}_t^-))\right) + N\gamma + K,$$

where K accounts for the (fixed) contribution of B_0, λ_0 and λ_1 to $\mathrm{CLL}(\mathcal{M} \mid T)$. Observe that $|I_0| + |I_1| = |T| = N$. Then, we define the *mixture-based factorized conditional log-likelihood* (mfCLL) scoring criterion as

$$\mathrm{mfCLL}(\mathcal{M} \mid T) = \sum_{i=1}^{n} \sum_{j=1}^{q_i^1} \sum_{k=1}^{r_i} (\alpha N_{ij1k}^1 + \beta N_{ij0k}^1) \log(\theta_{ij1k}). \tag{5.4}$$

By plugging in the OFE estimates in Equation (5.1) into the mfCLL criterion, we obtain

$$\mathrm{m\hat{f}CLL}(G_1 \mid T) = \sum_{i=1}^{n} \sum_{j=1}^{q_i^1} \sum_{k=1}^{r_i} (\alpha N_{ij1k}^1 + \beta N_{ij0k}^1) \log\left(\frac{N_{ij1k}^1}{N_{ij1}^1}\right), \tag{5.5}$$

where G_1 is the DAG of the foreground mixture component B_1. Observe that N^1_{ij1k} might be zero while $N^1_{ij0k} > 0$ which leads to an indeterminacy in Equation (5.5). To avoid this shortcoming pseudo-counts are commonly used. In practice, the use of pseudo-counts with the $\widehat{\mathrm{mfCLL}}$ scoring criterion turned out to be good for classification tasks (in opposition to $\hat{\mathrm{a}}$CLL presented in Section 4.3 at page 62). Intuitively, the reason for the good behavior of $\widehat{\mathrm{mfCLL}}$ is presumably due the fact that even if $N^1_{ij0k} > 0$ and $N^1_{ij1k} = 0$ we also have that $N^1_{ij1} \approx 0$ (because we have very few motif occurrences and they are moderately conserved), hence, by using pseudo-counts we have that $\frac{N^1_{ij1k}}{N^1_{ij1}} \approx \frac{1}{|\Sigma|} = \frac{1}{4}$. So, the logarithm in Equation (5.5) will not explode.

A MDL penalized version of the $\widehat{\mathrm{mfCLL}}$ score, called $\widehat{\mathrm{mfCLL}} - \mathrm{MDL}$, can be straightforwardly devised by subtracting the penalty

$$\frac{1}{2}\ln(N)|B_1| = \sum_{i=1}^{n} \frac{1}{2}\ln(N)(r_i - 1)q^1_i$$

to the $\widehat{\mathrm{mfCLL}}$ expression in Equation (5.5).

We are now able to introduce the algorithm that discriminatively learns the two-component mixture $\mathcal{M} = \langle \lambda_0, B_0, B_1 \rangle$ that maximizes (i) $\mathrm{LL}(\lambda \mid T)$, the log-likelihood of the mixing proportions $\lambda = \langle \lambda_0, \lambda_1 \rangle$; (ii) $\mathrm{MDL}(B_0 \mid T_0)$, the minimum description length of the background model B_0; and (iii) $\widehat{\mathrm{mfCLL}} - \mathrm{MDL}(B_1 \mid T)$, the $\widehat{\mathrm{mfCLL}}$ score with a MDL penalty of the foreground model B_1. The learning procedure relies on Algorithm 5.1 introduced in Section 5.3 (page 99) to compute each mixture component B_0 and B_1 and it is presented in Algorithm 5.3. Observe that other structures besides CκG can be learned (e.g. TAN) for each mixture component B_0 and B_1.

Algorithm 5.3 Learning CκG mixture models for binary classification tasks

```
1. Compute the mixing proportions $\lambda_0 = \frac{N_0}{N}$ and $\lambda_1 = 1 - \lambda_0$.
2. Learn generatively from $T_0$, using Algorithm 5.1 for the MDL score, the C$\kappa$G Bayesian network $B_0$.
3. Learn discriminatively from $T$, using Algorithm 5.1 for the $\widehat{\mathrm{mfCLL}} - \mathrm{MDL}$ score, the C$\kappa$G Bayesian network $B_1$.
```

5.4 Experimental results

To evaluate the ability of representing TFBS's with mixtures of C2G models we extracted 89 datasets of aligned binding sites from the TRANSFAC database (Wingender et al., 2001) for which there were 20 or more sites. These sequence-sets were used to build a motif model B_1. As background model we used 1000 sequences taken from promoter regions of the organism used to build B_1, resulting in a corresponding model B_0. In our experiments we used $k = 2$ which turned out to be a good tradeoff between efficiency and search space. Moreover, in order to avoid overfitting, that arises naturally when complex structures are searched, we endowed the intended scores with the MDL penalty. For each dataset we evaluated the ability of some relevant two-component mixtures pairs $B_0 - B1$, namely, TAN$-$TAN, TAN$-$C2G, C2G$-$TAN and C2G$-$C2G, to describe the distribution underlying the promoter regions of TFBS's. These mixtures were tested with $\mathrm{m\hat{f}CLL}$, with and without MDL penalty. Moreover, we also evaluate them with LL and MDL scoring criteria, as proposed by Barash et al. (2003).

We performed *5-fold cross-validation* tests in each dataset and conclude that discriminative mixtures of C2G models significantly outperformed the remaining mixtures learned both generative and discriminatively. Results are depicted in Table 5.3. Each entry of the table gives the Z-test and p-value of the significance test for the corresponding pairs of classifiers. The arrow points to the superior learning algorithm, in terms of classification rate. A double arrow is used if the difference is significant with p-value smaller than 0.05. In addition, scatter plots of the accuracies of the proposed methods against the others are depicted in Figure 5.3 (page 111). Points above the diagonal line represent cases where the method shown in the vertical axis performs better than the one on the horizontal axis.

Classifier **Struct.** **Param.**	TAN C2G $\mathrm{m\hat{f}CLL} - \mathrm{MDL}$	C2G C2G $\mathrm{m\hat{f}CLL}$	TAN C2G $\mathrm{m\hat{f}CLL}$	C2G TAN $\mathrm{m\hat{f}CLL} - \mathrm{MDL}$	C2G TAN $\mathrm{m\hat{f}CLL}$	TAN TAN MDL	TAN TAN LL
C2G C2G $\mathrm{m\hat{f}CLL} - \mathrm{MDL}$	1.94 0.03 $\Leftarrow$	8.16 $\ll 0.01$ $\Leftarrow$	5.12 $\ll 0.01$ $\Leftarrow$	2.74 < 0.01 $\Leftarrow$	6.05 $\ll 0.01$ $\Leftarrow$	6.05 $\ll 0.01$ $\Leftarrow$	7.71 $\ll 0.01$ $\Leftarrow$
TAN C2G $\mathrm{m\hat{f}CLL} - \mathrm{MDL}$		8.01 $\ll 0.01$ $\Leftarrow$	4.31 $\ll 0.01$ $\Leftarrow$	2.07 0.02 $\Leftarrow$	5.73 $\ll 0.01$ $\Leftarrow$	5.73 $\ll 0.01$ $\Leftarrow$	7.55 $\ll 0.01$ $\Leftarrow$
C2G C2G $\mathrm{m\hat{f}CLL}$			-7.99 $\ll 0.01$ $\Uparrow$	-7.99 $\ll 0.01$ $\Uparrow$	-7.62 $\ll 0.01$ $\Uparrow$	-7.62 $\ll 0.01$ $\Uparrow$	-7.09 $\ll 0.01$ $\Uparrow$
TAN C2G $\mathrm{m\hat{f}CLL}$				-4.55 $\ll 0.01$ $\Uparrow$	4.90 $\ll 0.01$ $\Leftarrow$	2.20 0.01 $\Leftarrow$	2.20 0.01 $\Leftarrow$

Table 5.3: Statistical significance of the results according to the Wilcoxon signed-rank test.

From Table 5.3 it is clear that C2G$-$C2G$-\mathrm{m\hat{f}CLL}$ is overfitting. Moreover, the same is not true, at least at the same scale, with TAN−C2G−m$\hat{\text{f}}$CLL. This points out that overfitting is mainly occurring in the background model B_0. Moreover, despite the fact that TAN−C2G−m$\hat{\text{f}}$CLL performed better than C2G−C2G−m$\hat{\text{f}}$CLL, Table 5.3 demonstrates that higher accuracies are achieved with the similar two-component models but with penalized versions of m$\hat{\text{f}}$CLL. Actually, the combination of m$\hat{\text{f}}$CLL − MDL scoring criterion with two-component mixtures of C2G models (C2G−C2G−m$\hat{\text{f}}$CLL−MDL in the fisrt line of Table 5.3) performs better than all the other considered classifiers. We conclude that discriminative learning of two-component mixtures of C2G Bayesian networks is beneficial, specially when the richness of the structure is controlled using MDL to avoid overfitting.

We also directly compared TAN−TAN−LL with TAN−TAN−m$\hat{\text{f}}$CLL classifiers, as well as TAN−TAN−MDL with TAN−TAN−m$\hat{\text{f}}$CLL − MDL, in order to understand the benefit of using the m$\hat{\text{f}}$CLL score without the noise introduced by the C2G model. Results show that TAN−TAN−m$\hat{\text{f}}$CLL significantly outperformed TAN−TAN−LL with a Z-score of 4.60362 leading to a p-value < 0.00003. Moreover, TAN−TAN−m$\hat{\text{f}}$CLL − MDL also performed significantly better than TAN−TAN−MDL with a Z-score of 4.68904 leading to a p-value < 0.00003. Furthermore, C2G−C2G−m$\hat{\text{f}}$CLL − MDL outperformed with statistical significance both TAN−TAN−m$\hat{\text{f}}$CLL and TAN−TAN−m$\hat{\text{f}}$CLL − MDL, both with a p-value < 0.00003. We conclude that a discriminative scoring criterion such as m$\hat{\text{f}}$CLL, with or without MDL penalty, is advantageous, when compared to their generative versions (as LL or MDL), in classification tasks. In addition, C2G−C2G−m$\hat{\text{f}}$CLL − MDL was the multinet classifier that performed the best.

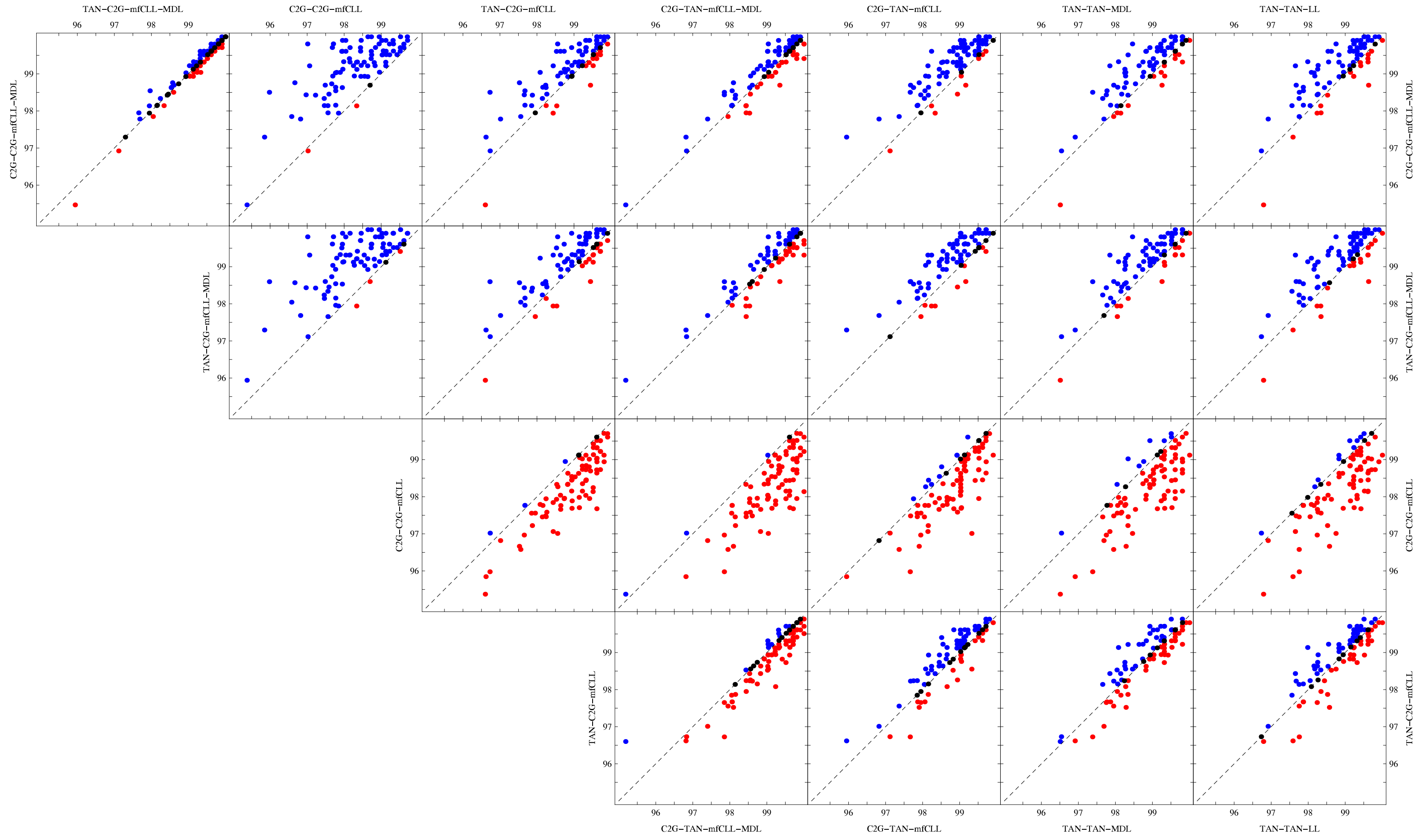


Figure 5.3: Scatter plots of the accuracy of different multinet classifiers.

Part III

Motif discovery

Chapter 6

RISOTTO: Improving RISO with maximal extensibility

The best known exact algorithms for the extraction of single (Sagot, 1998) and structured (Carvalho, Freitas, Oliveira, and Sagot, 2006) motifs perform well when searching for short motifs. In this chapter, we propose an improvement to such algorithms in order to deal with long motifs. The problem of extracting long motifs was first addressed by Pevzner and Sze (2000). They considered a precise version of the motif discovery problem: find all single motifs of length 15 with at most 4 mismatches in 20 sequences of size 600. In consequence several algorithms appeared (Pevzner and Sze, 2000; Buhler and Tompa, 2002; Eskin and Pevzner, 2002; Satya and Mukherjee, 2004).

A general solution for this problem deserves attention from the algorithmic point of view because its computational complexity is in the worst case exponential with respect to the number e of mismatches allowed among different occurrences of the same motif. The reason is that, to identify motifs of the required length, there can be an explosion in the number of candidates of intermediate length whose extension has to be attempted. This imposes, in practice, a limit to the length of the motifs, as in many applications the value of e depends on this length. The improvement introduced in this chapter acts exactly in these cases, turning possible the detection of relatively long motifs in practice.

The chapter is organized as follows. In Section 6.1 we present the single motif discovery algorithm followed by the structured motif discovery one in Section 6.2. These sections include the complexity analysis that compare the proposed algorithms with previously established ones. In Section 6.3 we present experimental results as well as a discussion about implementation issues.

6.1 Single motif extraction

The *single motif extraction problem*, presented in Section 3.2.2 (page 41), takes as input N sequences, a quorum $q \leq N$, a maximal number e of mismatches allowed, and a minimal and maximal length for the motifs, k_{min} and k_{max}, respectively (refer to Table 6.1). The problem consists in determining all motifs that e-occur in at least q of the N input sequences. Such motifs are called *valid models*.

For clarity purposes we need to abstract the details presented in the SPELLER algorithm (Algorithm 3.7, page 45). The simplified version is presented in Algorithm 6.1, where motif m is the one whose extension is being tried.

Algorithm 6.1 SPELLER, single motif extraction (simplified version)

```
SPELLER(motif $m$)
1. for each symbol $\alpha$ in $\Sigma$ do
2.   if $m\alpha$ is valid then
3.     if $|m\alpha| \geq k_{min}$ then spell out the valid model
4.     if $|m\alpha| < k_{max}$ then SPELLER($m\alpha$)
```

At the beginning SPELLER is called on the empty word. The algorithm recursively calls itself for longer motifs built by adding letters (Step 4), and considers new ones (Step 1) when the extension fails (Step 2). A valid motif is spelled out whenever a motif whose length lies within the required minimal and maximal length is being considered (Step 3). The order in which motifs are generated corresponds to a depth-first visit of a complete trie $\mathcal{M}$ of all words of length k_{max} over the alphabet Σ. We refer to $\mathcal{M}$ as the *motif tree*. In fact, the algorithm does not need to allocate the motif tree. The only memory requirement is for the suffix tree $\mathcal{T}$.

6.1.1 Using maximal extensibility of factors

The modification we suggest consists in storing information concerning *maximal extensibility* in order to avoid trying to extend hopeless motifs. For instance, assume that in our depth-first

Symbol	**Meaning**
Σ	alphabet (usually DNA or IUPAC)
α	symbol of the alphabet Σ
N	number of input sequences
n	average size of input sequences
e	number of mismatches allowed in a single motif
k_{min}	minimum motif size of a single motif
k_{max}	maximum motif size of single motif
q	quorum, i.e., number of sequences where the motif has to e-occur
m	potential motif
$\mathcal{M}$	(virtual) motif trie
$\mathcal{T}$	suffix tree of T
$MaxExt(m)$	maximal extensibility of m
$\lvert m\rvert$	length of m
$\langle m\rvert_k$	prefix of length k of m
$\lvert m\rangle_k$	suffix of length $\lvert m\rvert - k + 1$ of m
λ	empty word
p	number of boxes in structured motif extraction
e_i	number of mismatches allowed in the i-th box of a structured motif
k_{min_i}	minimum size of the i-th box of a structured motif
k_{max_i}	maximum size of the i-th box of a structured motif
d_{min_i}	minimum distance between the i-th and the $(i+1)$-box
d_{max_i}	maximum distance between the i-th and the $(i+1)$-box

Table 6.1: Definition of terms used in Chapter 6.

visit of the (virtual) motif tree $\mathcal{M}$, we have found out that motif m can be further extended without losing the quorum up to a length of $MaxExt(m)$ only, the latter representing its maximal extensibility. If later on, we are processing a motif m' that has m as a suffix, then the $MaxExt(m)$ information could be useful, as it applies to m' as well because m' can also be extended with at most $MaxExt(m)$ symbols (and possibly less). In particular, we have that if

$$|m'| + MaxExt(m) < k_{min},$$

then we can avoid any further attempt to extend m' because there is no hope to reach length k_{min} for motifs that have m' as prefix. Figure 6.1 illustrates exactly this example, that is, the extension of m' can be avoided, using $MaxExt(m)$, where m is a suffix of m', because $|m'| + MaxExt(m) < k_{min}$.

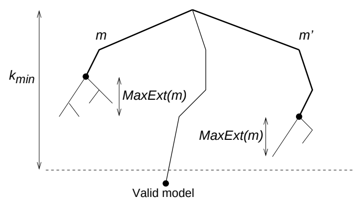


Figure 6.1: Example where the extension of m' can be avoided.

In order to understand how maximal extensibility is going to be used in motif extraction, notice that, in Algorithm 6.1, motifs are considered in lexicographical order by a depth-first visit of the (virtual) motif tree $\mathcal{M}$. Every time we stop extending a motif, that is, when we backtrack in $\mathcal{M}$, it is either because we found a valid motif of the maximal length, or because the quorum is no longer satisfied ($m\alpha$ does not satisfy the condition at Step 2, and we start to consider the next one in lexicographical order). More rigorously, the analysis of the motif $m = \sigma_1, \ldots, \sigma_{|m|}$ with $\sigma_i \in \Sigma$, for all $i = 1, \ldots, |m|$, is abandoned either when m is valid and $|m| = k_{max}$, or m does not satisfy the quorum.

In the first case, m is valid, as are all its prefixes, and $|m| = k_{max}$. No information on the maximal extension of m nor of its prefixes can be of any use because all motifs having a prefix of m as suffix can in general still be extended as much as necessary to reach at least the length k_{min}. For this reason, we set $MaxExt(m) = +\infty$, meaning that m can be extended possibly more than we are computing.

In the second case, m does not satisfy the quorum while all its prefixes do. For reasons that will be clearer later, we chose to only use the maximal extensibility information of motifs of length up to $k_{min} - 1$, hence this case can be subdivided into two sub-cases. When a motif m cannot be extended anymore and it has not reached the length $k_{min} - 1$, we set $MaxExt(m) = 0$. If the motif has reached a length h between $k_{min} - 1$ and k_{max}, we set $MaxExt(\langle m\alpha|_{k_{min}-1}) = h - (k_{min} - 1)$, where $\langle m\alpha|_{k_{min}-1}$ is the prefix of length $k_{min} - 1$ of $m\alpha$. Since it can be that $MaxExt(\langle m\alpha|_{k_{min}-1})$ had already received some value because a previous extension of $\langle m\alpha|_{k_{min}-1}$ was interrupted, then we change the value of $MaxExt(\langle m\alpha|_{k_{min}-1})$ only if we are increasing it, as maximal extensibility of a motif refers to its longest extension. We assume that all maximal extensibility values are initially set to -1, hence the first attribution to $MaxExt(\langle m\alpha|_{k_{min}-1})$ will always increase its value.

In all aforementioned cases, the algorithm does not consider any further extension of m, and backtracks. This backtracking consists in either replacing the last letter $\sigma_{|m|}$ of m (Step 1), or considering a shorter motif which in general shares a prefix with m, if $\sigma_{|m|}$ was the last letter of the alphabet Σ. In this latter case, the whole subtree rooted at the node spelling $\sigma_1 \ldots \sigma_{|m|-1}$ has been completely visited. Thus, we have all the information necessary to set the value of $MaxExt(\sigma_1 \ldots \sigma_{|m|-1})$ according to $MaxExt(x) = 1 + \max_{\alpha\in\Sigma} MaxExt(x\alpha)$, for all valid motifs x such that $|x| < k_{min} - 1$. If the letter $\sigma_{|m|-1}$ was the last of the alphabet, then the backtracking goes further. In that case, also the $MaxExt$ information concerning the word $\sigma_1 \ldots \sigma_{|m|-2}$ can be filled in the same way, and so on as long as we climb up in the tree.

As mentioned before, maximal extensibility information can be used for motifs whose extension is being considered and for which this information could actually prevent some useless attempts. Namely, assume we are trying to extend the motif $m = \sigma_1, \sigma_2 \ldots, \sigma_{|m|}$. Since the motifs are considered by means of a depth-first search on the virtual motif tree, we obviously do not know the value of $MaxExt(m)$ yet. Moreover, we know $MaxExt(\sigma_2, \ldots, \sigma_{|m|})$ only if it lexicographically precedes m, that is, it has already been visited in the motif tree. If this is not the case, we check whether $MaxExt(\sigma_3, \ldots, \sigma_{|m|})$ is already known, and so on, possibly until the singleton $\sigma_{|m|}$. If they are all lexicographically greater than m, then no maximal extension information can be used for m, but if for any of them $MaxExt$ is known and it holds that the maximal possible extension is not enough to reach k_{min}, then the information is useful as it guarantees that attempting to further extend m is useless.

Lemma 6.1.1 Let $w \in \Sigma^*$. We have $MaxExt(w) \leq MaxExt(v)$ for each v which is a suffix of w.

Proof: Let $MaxExt(w) = k$. There exists $s \in \Sigma^k$ such that the motif ws is valid, that is, it appears in at least q sequences, and no longer string in Σ^* has the same property. Let us now assume that there is a suffix v of w such that $MaxExt(v) = j < k$. Then there exists $t \in \Sigma^j$ with $j < k$, and no longer t, such that the motif vt is valid. However, we know that there exists $s \in \Sigma^k$ such that ws appears in at least q sequences. Since vs is a suffix of ws, and since it satisfies the quorum, then the hypothesis is contradicted. $\square$

A consequence of Lemma 6.1.1 is that longer suffixes of m can give us more tight bounds on the maximal extensibility information with respect to shorter ones. Therefore, since we start by checking the longest one, as soon as we find a suffix of m that enables us to state that m is not worth further extension attempts, then we can stop checking the other (shorter) suffixes. That is, if we find a suffix $|m\rangle_j = \sigma_j, \ldots, \sigma_{|m|}$ of m, with $1 < j \leq |m|$, such that $MaxExt(|m\rangle_j)$ is not enough for m to reach k_{min} because $MaxExt(|m\rangle_j) + |m| < k_{min}$, then we can quit attempting m and all its extensions, and we can consequently update $MaxExt(m)$. On the other hand, if no suffix $|m\rangle_j$ of m is such that $MaxExt(|m\rangle_j) + |m| < k_{min}$, then the maximal extension does not disallow to reach k_{min}. In this case, we have to go on trying to extend m even if it might be the case that it will never reach the minimal length.

The algorithm for single motif extraction using the maximal extensibility information, called RISOTTO, is presented in Algorithm 6.2. For simplicity, we denote in the same way a node x and the word spelled by the path from the root to x. Moreover, recall that we use $\langle m\alpha|_{k_{min}-1}$ to denote the prefix of $m\alpha$ of length $k_{min} - 1$, and $|x\rangle_{|x|-1}$ to denote the suffix of x of length $|x| - 1$. Finally, with regard to Step 3, recall that we assumed that all maximal extensibility values are initially set to -1.

6.1.2 Complexity analysis

The time complexity of Algorithm 6.2 remains the same as for Algorithm 6.1 in the worst case. Nevertheless, the proposed improvement has very positive effects on the average case. Next, we show how to compute, in average, the ratio between the number of attempted extensions by RISO and RISOTTO for single motif extraction and compute the limit from which RISOTTO performs better than RISO.

Algorithm 6.2 RISOTTO, single motif extraction with maximal extensibility

```
RISOTTO(motif $m$)
1.  for each symbol $\alpha$ in $\Sigma$ do
2.    $x := m\alpha$
3.    repeat $x := |x\rangle_{|x|-1}$ until ($x = root$ or $MaxExt(x) \neq -1$)
4.    if $x \neq root$ and $MaxExt(x) + |m\alpha| < k_{min}$ then
5.      $MaxExt(m\alpha) := MaxExt(x)$
6.      stop spelling $m\alpha$ and continue
7.    if $m\alpha$ is valid then
8.      if $|m\alpha| \geq k_{min}$ then spell out the valid model
9.      if $|m\alpha| < k_{max}$ then RISOTTO($m\alpha$)
10.     else $MaxExt(\langle m\alpha|_{k_{min}-1}) := +\infty$
11.   else
12.     if $|m\alpha| < k_{min}$ then $MaxExt(m\alpha) := 0$
13.     else if $MaxExt(\langle m\alpha|_{k_{min}-1}) < |m\alpha| - (k_{min} - 1)$ then $MaxExt(\langle m\alpha|_{k_{min}-1}) := |m\alpha| - (k_{min} - 1)$
14. if $|m| < (k_{min} - 1)$ then $MaxExt(m) := 1 + \max_{\alpha \in \Sigma} MaxExt(m\alpha)$
```

Assume that the dataset has r planted random motifs of size t, where each motif can be extracted with at most e mismatches, and that the remaining text is uniformly random. This assumption captures the fact that we want to analyze the ratio between the number of attempted extensions by RISO and RISOTTO in the context of a dataset with highly correlated sequences (meeting the application requirements to biological datasets).

Let M_i be the random variable that gives the number of valid motifs of size i with at most e mismatches for the assumed dataset, where $0 \leq i \leq t$. Clearly, we have that $P(M_0 = 1) = 1$ and $P(M_t \geq r) = 1$. The number of attempted extensions by RISO at level $i > 0$ (when the recursion step is at level i) is given by the random variable

$$E_i = M_{i-1}|\Sigma|,$$

and the total number of attempted extensions for the extraction of a single motif of size k is given by

$$R_k = \sum_{i=1}^{k} E_i.$$

On the other hand, RISOTTO will only extend words at level i if they fulfill the maximum extensibility requirement. Therefore the number of attempted extensions by RISOTTO at level i is given by

$$E'_i = M_{i-1}|\Sigma|(1 - p(i)),$$

where $p(i)$ is the (random variable denoting the) probability of a i-word having maximal extensibility information to avoid its extension. Furthermore, the total number of attempted extensions by RISOTTO for the extraction of a single motif of size k is given by

$$R'_k = \sum_{i=1}^{k} E'_i.$$

We conclude that to compute the ratio of the means of R'_k and R_k, that is,

$$\frac{\overline{R'_k}}{\overline{R_k}}$$

we need to determine the means of the random variables M_i and $p(i)$, for $i = 1, \ldots, k$. We proceed by computing the mean of M_i. Clearly, a planted motif of size t has $t-i+1$ segments of size i (considering overlapping). Observe that the average number of mismatches of the e-occurrences of an extracted motif of size t (recall $\nu(e,t)$ in page 44) is given by

$$\overline{e} = \sum_{j=0}^{e} j \frac{\begin{pmatrix} t \\ j \end{pmatrix} (|\Sigma| - 1)^j}{\nu(e,t)}.$$

Hence, if we assume the mismatches to distribute uniformly over the segments, the average number of mismatches of the segments of size i of the e-occurrences is

$$\overline{e}_i = \frac{i}{t}\overline{e}.$$

Thus, the motifs extracted at level i due to the planted motifs are all the neighbors differing at most $(e - \overline{e}_i)$ letters from the segments of size i of the planted motifs. Since there are $r(t-i+1)$ segments of size i, and assuming that these segments are different, the average number of extracted motifs of size i with at most e mismatches due to the planted motifs is

$$\overline{T}_i = |\Sigma|^i \left(\sum_{j=0}^{r(t-i+1)-1} \left(1 - \frac{\nu(e-\overline{e}_i, i)}{|\Sigma|^i}\right)^j \frac{\nu(e-\overline{e}_i, i)}{|\Sigma|^i} \right).$$

Finally, to determine the mean of M_i, we need to take into account the motifs extracted from the random part of the text, and so, we have

$$\overline{M}_i = \overline{T}_i + (|\Sigma|^i - \overline{T}_i)(1 - \pi_i)$$

where π_i is the probability of a random word of size i not being extracted with quorum q from a set of N sequences. Given that the probability of an e-neighbor of a word of size i not appearing in a random text of size n is

$$\delta(i,e,n) = \left(1 - \frac{1}{|\Sigma|^i}\right)^{(n-i+1)\nu(e,i)} \approx \left(1 - \frac{1}{|\Sigma|^i}\right)^{ni^e|\Sigma|^e},$$

the value of π_i can be computed by the following binomial

$$\pi_i = \sum_{j=0}^{q-1} \begin{pmatrix} N \\ j \end{pmatrix} \delta(i,e,n)^{N-j}(1-\delta(i,e,n))^j .$$

We finalize by computing the (expected value of) probability $p(i)$. Since the probability of a suffix of a random word being lexicographically smaller than the random word is $\frac{1}{2}$, we have that

$$p(i) = \sum_{j=1}^{i-1} \frac{1}{2^j}\gamma_{i-j}$$

where γ_{i-j} is the probability of the suffix of size $i-j$ to have information to avoid the extension. Notice that γ_{i-j} is the probability of the suffix of size $i-j$ not being extended to a size greater than $k-1$, and is given by

$$\begin{aligned}\gamma_{i-j} &= \pi_{i-j} + (1-\pi_{i-j})\pi_{i-j+1}^{|\Sigma|} + (1-\pi_{i-j})(1-\pi_{i-j+1}^{|\Sigma|})\pi_{i-j+2}^{|\Sigma|^2} + ... \\ &= \sum_{v=0}^{k-1-(i-j)} \pi_{i-j+v}^{|\Sigma|^v} \prod_{\ell=0}^{v-1}(1-\pi_{i-j+\ell}^{|\Sigma|^\ell}) .\end{aligned}$$

To understand when RISOTTO starts to provides a gain over RISO, it is important to look to E'_i and E_i. Note that if M_{i-1} is larger than M_i, E'_i will be much smaller than E_i if $p(i)$ is close to 1. Moreover, as soon as random motifs start to disappear, M_{i-1} will be larger than M_i, which happens when π_i is close to 1. Both π_i and $p(i)$ depend tightly of $\delta(i,e,n)$, that is, if $\delta(i,e,n)$ is close to 0, so are π_i and $p(i)$, and if $\delta(i,e,n)$ is close to 1, so are π_i and $p(i)$. Since $1-\delta(i,e,n)$ behaves like a Dirac cumulative function (in i) for large values of n, that is, it jumps very fast from 0 to 1, we just need to solve the equation $\delta(i,e,n) = 1/2$ for the variable i to grasp when RISOTTO starts to be faster than RISO, which is just slightly before the solution. The solution of that equation is the fixed point of the following function

$$f(x) = -\frac{\log\left(1 - 2^{-\frac{1}{|\Sigma|^e x^e n}}\right)}{\log(\Sigma)}.$$

Given that $f(x)$ is contractive, that is, its derivative function takes values in the interval $(-1,1)$, the fixed point can be computed by iterating f over an initial value. Finally, notice that the fixed point increases with the values of e, n and Σ.

With the previous analysis, we have all the machinery necessary for computing the ratio between the expected number of attempted extensions between RISO and RISOTTO, as well as, from which point RISOTTO performs better than RISO. As an example, the ratio between the expected number of extensions attempted by RISOTTO and RISO for a dataset consisting of $N = 100$ sequences of size $n = 1000$ where we planted $r = 1$ motif of size $t = k = 5..20$, with up to $e = 2$ mismatches, and quorum $q = 100$, is given in Figure 6.2. For the dataset considered, the fixed point for $f(x)$ is $x = 10.6616$.

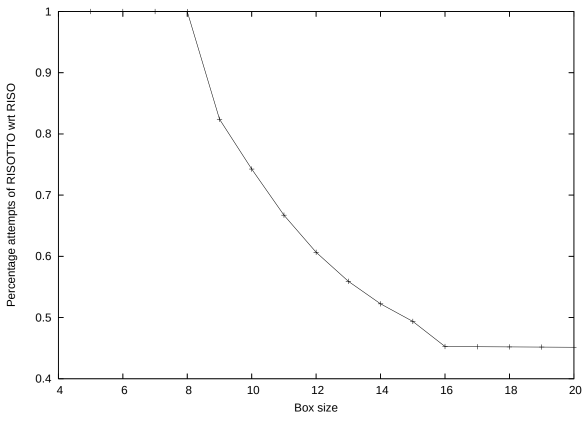


Figure 6.2: Ratio between the expected number of extensions attempted by RISOTTO and RISO.

6.2 Structured motif extraction

The *structured motif extraction problem*, presented in Section 3.2.3 (page 45), takes as parameters N input sequences, a quorum $q \leq N$, p maximal error rates $(e_i)_{i \leq 1 \leq p}$ (one for each of the p boxes), p minimal and maximal lengths $(k_{min_i})_{i \leq 1 \leq p}$ and $(k_{max_i})_{i \leq 1 \leq p}$ (one for each of the p boxes), and $p - 1$ intervals of distance $(d_{min_i}, d_{max_i})_{i \leq 1 \leq p-1}$ (one for each pair of consecutive boxes). Given these parameters, the problem consists in searching for the contents of the boxes, that is the motifs, that have the structure defined by the parameters above and that satisfy the quorum.

For clarity purposes we need to abstract the details presented in RISO algorithm presented in Algorithm 3.8 (page 49). The simplified version is presented in Algorithm 6.3, where motif m is the one whose extension is being tried. Herein, we assume a structured motif composed by two boxes only ($p = 2$). The modifications we suggest in this work can be extended to all the more complex variants of the problem, as they do not depend on whether p equals 2 or is bigger.

Algorithm 6.3 RISO, structured motif extraction (simplified version)

```
RISO(motif $m$, box $i$)
  1. for each symbol $\alpha$ in $\Sigma$ do
  2.   if $m\alpha$ is valid then
  3.     if $|m\alpha| \geq k_{min_i}$ then
  4.       if $i = 2$ then spell out the valid model
  5.       else follow box-links and update $\mathcal{T}$ to RISO($\lambda, 2$)
  6.     if $|m\alpha| < k_{max_i}$ then RISO($m\alpha, i$)
```

At the beginning RISO is called on the empty word and with $i = 1$. The algorithm recursively calls itself for longer motifs built by adding letters (Step 6), until possibly considering the second box (Step 5), and it considers new ones (Step 1) when the extension fails (Step 2). A valid motif is spelled out whenever a second box, whose length lies within the required minimal and maximal length, is being considered (Step 4). The update in $\mathcal{T}$ mentioned in Step 5 basically consists in a jump following the box-links, from the nodes reached by the first box, to reach potential end positions for the second boxes of the motif. The nodes reached in the jump are then modified with the information stored in the box-links. This information reflects the input sequences the jump concerns to and it is used to temporarily and partially modify $\mathcal{T}$. The extraction of the second box then proceeds in the same way over the modified part of the tree. Once the operation of extracting all valid motifs $\langle(m_1, m_2),(d_{min}, d_{max})\rangle$ has ended, $\mathcal{T}$ is restored to its previous state. The construction of another motif m_1 then follows and the process continues until all valid structured motifs are extracted.

6.2.1 Using maximal extensibility of factors

In the case of structured motifs, the maximal extensibility information for the first box of a motif should be updated as described in Section 6.1.1. However, any failure in attempting to extend a motif during the search of a second box cannot update any value of $MaxExt$ because it refers only to parts of the text that follow a specific first box at a specific distance. In fact, when a first box m_1 of a structured motif is fixed at any given step, the maximal extensibility information that concerns the whole sequence is in general an upper bound on the maximal extensibility of fragments of the sequence that are at a given distance from the occurrences of m_1. Given this observation, a possibility is to use the maximal extensibility information of the first box when searching and trying to extend a second box. Another possibility, while attempting to find a motif for the second box, is to compute and store tighter maximal extensibility information which we can use for the second box being attempted as long as the first box is fixed. In the following, we only address the first alternative, that is, only the first box stores extensibility information.

The conditions needed for our optimization to be applicable in the case of structured motifs may hold even more frequently than in the case of single motifs. In fact, since the search for a valid motif as second box is made after a valid motif for the first box is found, maximal extensibility information may be known also for the whole motif whose extension is attempted and not just for its prefixes. In other words, it may happen that when Algorithm 6.3 is called with parameters m and 2, the value of $MaxExt(m)$ is already known. Proper suffixes are thus not the only candidates to give useful information when we are trying to find a motif for the second box. The extensibility information can be used as for the case of single motifs except that one has to deal with different error rates among boxes. Indeed, e_2 must be less than or equal to e_1 in order for the extensibility information to be useful for the second box. Otherwise, the maximal extensibility information stored for the first box may be too restrictive, and if it is used, it may cancel the extension of valid motifs.

The algorithm for structured motif extraction using the maximal extensibility information is presented in Algorithm 6.4. Similarly to the case of single motif extraction, the time complexity of Algorithm 6.4 remains the same as for Algorithm 6.3 in the worst case, and the improvement proposed accounts only for the average case, as we shall verify in the next section.

6.3 Implementation and experimental results

In order to verify the improvement proposed over RISO, a C implementation of the maximal extensibility algorithm, called RISOTTO,[1] was made. The new implementation was tested against a C implementation of the RISO algorithm presented in Carvalho, Freitas, Oliveira, and Sagot (2005). The results of the experiments show a significant improvement for both single and structured motif extraction when using maximal extensibility information. As it

[1]RISOTTO is available at `http://kdbio.inesc-id.pt/~asmc/software/risotto.html`.

Algorithm 6.4 RISOTTO, structured motif extraction with maximal extensibility

```
RISOTTO(motif $m$, box $i$)
 1. for each symbol $\alpha$ in $\Sigma$ do
 2.   if $i = 1$ or $e_2 \leq e_1$ then
 3.     $x := m\alpha$
 4.     while ($x \neq root$ or $MaxExt(x) = -1$) do $x := |x\rangle_{|x|-1}$
 5.     if $x \neq root$ and $MaxExt(x) + |m\alpha| < k_{min_i}$ then
 6.       if $i = 1$ then $MaxExt(m\alpha) := MaxExt(x)$
 7.       stop spelling $m\alpha$ and continue
 8.   if $m\alpha$ is valid then
 9.     if $|m\alpha| \geq k_{min_i}$ then
10.       if $i = 2$ then spell out the valid model
11.       else follow box-links and update $\mathcal{T}$ to RISOTTO$(\lambda, 2)$
12.     if $|m\alpha| < k_{max_i}$ then RISOTTO$(m\alpha, i)$
13.     else if $i = 1$ then $MaxExt(\langle m\alpha|_{k_{min_1}-1}) := +\infty$
14.   else if $i = 1$ then
15.     if $|m\alpha| < k_{min_1}$ then $MaxExt(m\alpha) := 0$
16.     else if $MaxExt(\langle m\alpha|_{k_{min_1}-1}) < |m\alpha| - (k_{min_1} - 1)$ then
          $MaxExt(\langle m\alpha|_{k_{min_1}-1}) := |m\alpha| - (k_{min_1} - 1)$
17. if $i = 1$ and $|m| < (k_{min_1} - 1)$ then $MaxExt(m) := 1 + \max_{\alpha \in \Sigma} MaxExt(m\alpha)$
```

turns out, maximal extensibility may cost some extra space, which is a delicate issue for large datasets, but it can definitely save some hopeless visits, and in general it improves the efficiency of the search.

6.3.1 Storing the extensibility information

We start with some considerations concerning the storage of extensibility information. As we have seen in Section 6.1.1, due to the order in which motifs are considered, we have that only certain sub-words of motifs can give useful information concerning maximal extensibility, namely, those that are lexicographically smaller. Since no motif is smaller than itself, we actually only use the $MaxExt$ information of motifs that are shorter than the current one, that is, they are proper suffixes. Therefore, since the condition to check is whether or not we can hope to reach the k_{min} length, then we make use of the $MaxExt$ data only for strings of length at most $k_{min} - 1$. Hence, it is not necessary to store this information for motifs that have length k_{min} or more for the purpose mentioned above.

Let us now discuss how much space is required to store the extensibility information until level $k_{min} - 1$. We say that a tree is *uncompact complete* if it is a trie where all possible nodes are present. There is thus no arc whose label contains more than one letter. A previous result from Allali (2000) makes use of some statistical analysis for stating that a suffix tree of a text of length n is expected to be uncompact complete at the $log_{|\Sigma|}(n)$ top levels, where Σ is the alphabet of the text. This fact suggests a model to store extensibility information: a static data structure to keep the $MaxExt$ values until level $log_{|\Sigma|}(n)$, and a dynamic structure for deeper levels. Since we are interested in the DNA alphabet (composed of the four nucleotides A, C, G, and T), then we have that our suffix tree is uncompact complete at the top $log_4(n)$ levels where n is the size of the input sequence s. The function $log_4(n)$ reaches 10 for $n \approx 10^6$, it is greater than 11 for $n = 10^7$, it is more than 13 for $n = 10^8$, and nearly 15 for $n = 10^9$. These values correspond to reasonable values for the minimal length k_{min} of the motif, and they are reached for values n of the text size corresponding to quite big datasets.

In the RISOTTO implementation, we took all the aforementioned observations into consideration. Since k_{min} has to be relatively small for our approach to be tractable space-wise, we considered only 1 byte (a char in C) to store $MaxExt$ values. In this case, extensibility values must be less than 256, which is quite reasonable. To build a static data structure to store such values until level z, we need $z + 1$ 1-byte arrays, where the j-th array has size $|\Sigma|^j$ with $0 \leq j \leq z$. Therefore, for the case of DNA, the total amount of memory required is $\frac{4^{z+1}-1}{3}$ bytes. This function gives us values of 1.3MB for $z = 10$, 5.3MB for $z = 11$, 85.3MB for $z = 13$, and 1.3GB for $z = 15$.

In our experiments, we achieved an optimum trade-off between memory allocation versus management and maximal extensibility gain when $z = 10$. Taking this observation into account, we only allocate values for $MaxExt$ until level $z = \min\{10, k_{min} - 1\}$, even for large values of k_{min}, and disregard deeper levels as well as the dynamic data structure mentioned above. Nevertheless, we allowed this z level to be an implementation parameter. In the end, considering $z = \min\{10, k_{min} - 1\}$, RISOTTO requires at most 1.3MB more that RISO for DNA databases, being more than twice as fast as we shall see next.

6.3.2 Experimental results

To test maximal extensibility performance we used several randomly generated (with a uniform distribution over the four letters of the DNA alphabet) synthetic datasets with planted structured motifs. Each dataset consists of 100 sequences of size 1000 where we planted one motif, possibly structured into several boxes, with 2 mismatches per box.

We ran both RISO and RISOTTO requiring a quorum $q = 100$ and at most 2 mismatches per box so that the output contains at least the planted motif. For each dataset, we made several runs for increasing lengths of the motifs. In particular, given the number of boxes of the structured motifs (in our tests there are p boxes for $p = 1, \ldots, 4$), we have increased the size of the boxes ranging from 5 to 20. As a result, the total motifs size (without counting the gaps) ranges from 5 to 80.

For each p (number of boxes), we have plotted in Figure 6.3, against the size of the motif (x axis), the ratio between the number of extensions attempted by RISOTTO and those by RISO (y axis). We refer the reader to Figure 6.2 at Section 6.1.2 (page 124) to compare theoretical with experimental results in Figure 6.3 (top left) obtained in the same dataset. Given than RISOTTO only saves useless attempts, this equals the percentage of saved calls

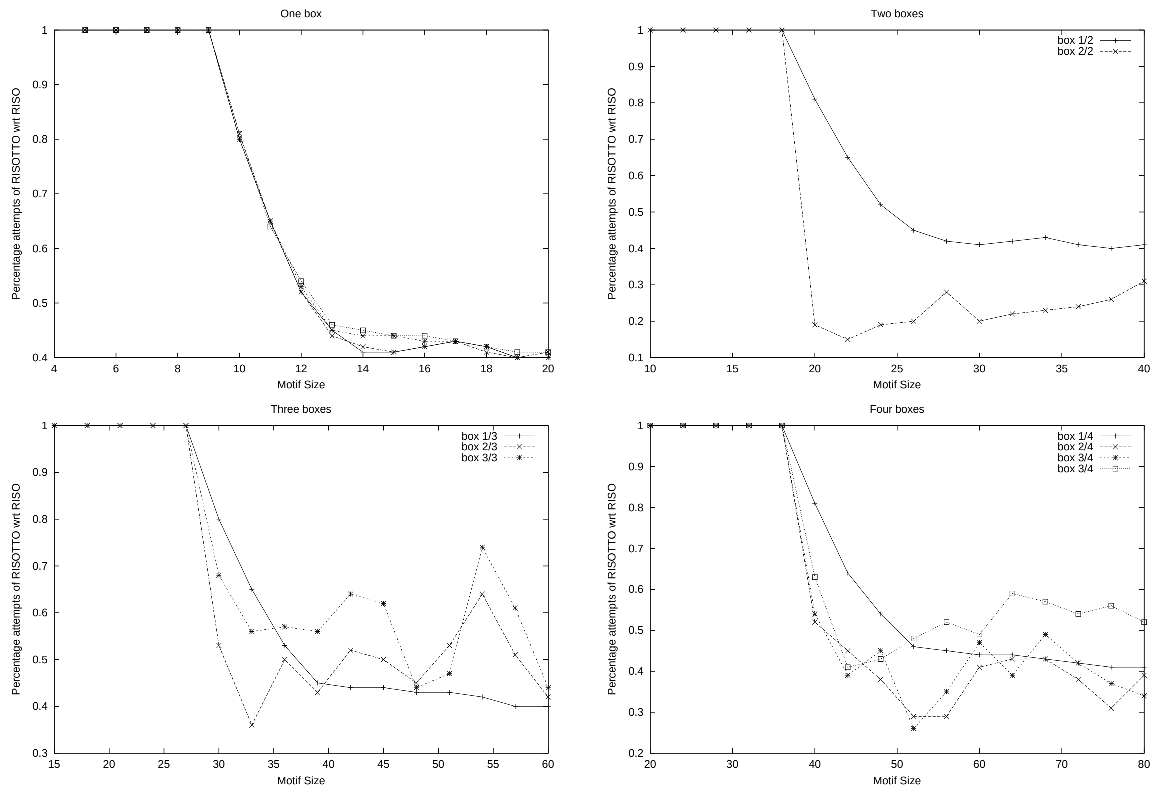


Figure 6.3: Ratio between the number of extensions attempted by RISOTTO and RISO.

of the recursive procedure. For one box (Figure 6.3 top left) we have depicted the results for several runs, while for two, three and four boxes (Figure 6.3 top right and bottom) we show only one curve for the inference of each box of the structured model.

As one would expect, the attempts saved are more when the length of the motif increases and, in particular, the improvement starts when the length of the box is about 10 (this value depends in general on the input sequence and the alphabet size). For one box (see Figure 6.3 top left), the number of attempted extension of RISOTTO decreases fast to 40% with respect to RISO (for growing values of the length of the motifs). Even better results, getting as good as attempting only 20% of the extensions of RISO, were achieved when extracting an i-th box with $2 \leq i \leq p$ (see Figure 6.3 top right and bottom). Moreover, we present the ratio of speed performance of the computation of RISOTTO with respect to that of RISO. This is shown for all tests together in Figure 6.4 for all possible sizes of the boxes. One can see that

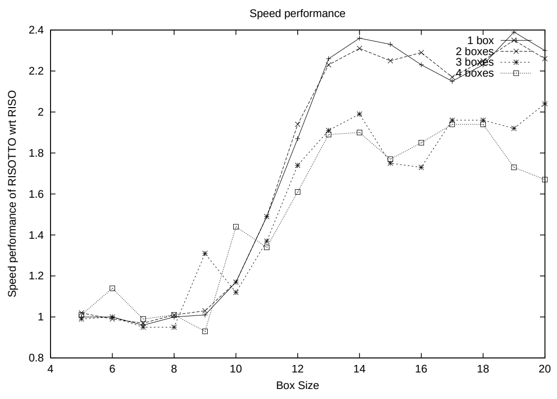


Figure 6.4: Ratio between performance of RISOTTO and RISO.

the best relative performance is achieved for the first boxes (that is where it is more needed because the search space is very large and noisy), where RISOTTO is up to 2.4 faster than RISO.

Finally, in Pevzner and Sze (2000) a challenging problem was launched that concerned finding all single motifs of length 15 with at most 4 mismatches in 20 texts of size 600. We ran both RISO and RISOTTO on such instances. We observe a speedup of 1.6 of RISOTTO over RISO. We actually believe that a true challenge should involve texts of larger size. Therefore, we ran tests with the same parameters (length 15 and at most 4 mismatches) on larger input sequences. The results confirm the 1.6 speedup for sequences of length 700 and 800, 1.3 speedup for length 900, and then the speedup decreases, but the time required by RISOTTO is always lower than for RISO.

Chapter 7

GRISOTTO: Improving RISOTTO with prior knowledge

Herein, we extend the RISOTTO combinatorial algorithm (Pisanti, Carvalho, Marsan, and Sagot, 2006), presented in the previous chapter, to take into account prior information. Since methods based on the detection of overrepresentation of TFBS's in co-regulated DNA sequences are known to face problems detecting weak motifs, we propose to post-process the RISOTTO output by modifying top motifs in a greedy fashion, guided by the information given by the prior. The rational for this approach is that the combinatorial algorithm exploits the full space of possible motifs pointing out good candidates. Afterwards a greedy search is performed over these initial guesses and good motifs are up-weighted by the prior. The reduction of the search space attained in the greedy search by using the output of a combinatorial algorithm makes the proposed algorithm, called GRISOTTO, very efficient.

A great advantage of GRISOTTO is its ability to combine priors from different sources. This is achieved by considering a convex combination of the information given by all priors resulting in an information-theoretical scoring criterion called *Balanced Information Score* (BIS). To unravel the benefits of using BIS with GRISOTTO we focus on discovering motifs in 156 benchmark datasets from ChIP-chip data from yeast. We considered three different priors already used by PRIORITY, namely, orthologous conservation (Gordân et al., 2008, 2010), DNA duplex stability (Gordân and Hartemink, 2008) and nucleosome positioning (Narlikar et al., 2007). By combining the information of these three priors together in BIS we guided the GRISOTTO greedy search and achieved considerably more accurate results than by using the priors separately. We further verified that GRISOTTO is at least as accurate as the PRIORITY and MEME (Bailey et al., 2010) algorithms when using the same priors separately.

We also gauge GRISOTTO with 13 sequence-sets from mouse ChiP-seq data. In this evaluation we used two different priors providing extra information from orthologous conservation (Bailey et al., 2010) and coverage profiles given by ChiP-seq assays (Hu et al., 2010). Results show that orthologous conservation was able to uncover motifs that resemble ones already reported by previous works on the same data (Chen et al., 2008; Bailey et al., 2010). However, the prior built from the ChiP-seq assays was not very beneficial to GRISOTTO, as it reported exactly the same motifs as the *uniform prior*. We attributed this to the fact that the information contained in this prior is already encoded in the BIS score. Indeed, coverage profiles indicate overrepresentation, expressed via high sequencing read density, and the BIS score is a weighted balance between overrepresentation and priors.

Besides effectiveness, GRISOTTO also showed to be very efficient, taking around 2 to 3 seconds per yeast sequence-set, that have around 200 sequences of 500bp, and 1 to 4 minutes per mouse sequence-set, that have from around 1000 to 40000 sequences of 200bp. These computational times were obtained using one core of an Intel 2.4 GHz core 2 Duo and include the generation of the initial starting points by RISOTTO. We conclude that post-processing the output of combinatorial algorithms guided with the information given by single or combined priors yields an efficient approach that shows great promise in extending the power of motif discovery tools.

The chapter is organized as follows. In Section 7.1 we present the GRISOTTO algorithm followed by the BIS scoring criterion in Section 7.2. In Section 7.3 we present results from ChiP-chip data and ChiP-seq data. Detailed setup, evaluation methodology and detailed results of GRISOTTO are presented in Appendixes B–D (page 163). Finally, we discuss in Section 7.4 some issues brought up during the development of GRISOTTO.

7.1 GRISOTTO algorithm

In this section we present the GRISOTTO algorithm for motif discovery. The proposed algorithm uses the RISOTTO (Pisanti, Carvalho, Marsan, and Sagot, 2006) output as starting points of a greedy procedure that aims at maximizing a scoring criterion based on combined prior information. Our approach diverges from EM (as in MEME) and Gibbs sampling (as in PRIORITY) as we do not consider latent variables and do not use a background model. Moreover, instead of maximizing the likelihood, we propose a scoring criterion based on the balanced information of observing the DNA sequences and the priors given a candidate motif. We called this score *Balanced Information Score* (BIS). Finally, instead of reporting a PSSM, GRISOTTO returns the IUPAC string that is best fitted, according to BIS, via a greedy search procedure.

We next introduce some notation needed to describe the GRISOTTO algorithm (refer to Table 7.1, page 134). Start by considering that we have a set of N co-regulated DNA sequences henceforward denoted by $f = (f_i)_{i=1,\ldots,N}$. The length of each sequence f_i is n_i, that is, $f_i = (f_{ij})_{j=1,\ldots,n_i}$. Moreover, prior information is resumed as a *position-specific prior* (PSP), where each position (i, j) in the prior amounts for the likelihood that a motif starts in the base f_{ij} of the sequence f_i from the set f of co-regulated DNA sequences. In that case, consider that S_p contains some prior information in a PSP format about the domain in study, with $p = 1 \ldots \ell$, where ℓ is the number of priors (eventually zero). We denote by $S = \langle S_1, \ldots, S_\ell \rangle$ the list of all priors. The goal of GRISOTTO is to report a single motif of a fixed size k, that is, an IUPAC string of size k. The IUPAC alphabet is henceforward denoted by Σ.

The pseudocode of GRISOTTO is depicted in Algorithm 7.1. The algorithm starts by running RISOTTO to extract, at least z_{min}, and at most z_{max}, motifs of size k (see details in Appendix B, page 163). From the RISOTTO output, the top z motifs are collected in a set called $\mathcal{C}$ (Step 2) and constitute the starting points of the GRISOTTO greedy procedure, called GGP (Step 4). Briefly, GGP starts with a motif $m \in \mathcal{C}$ and returns the best fitted motif, according to BIS, by updating each position in m with an IUPAC symbol until no local improvements can be achieved. In Step 5-6 the variable r, that stores the output of the algorithm, is updated whenever the GGP procedure returns a motif with a BIS score higher than the current stored one. Note that in Step 2 the result variable r is initialized with the empty motif ε. We consider that the empty motif ε has the minimum possible BIS value.

It remains to explain the GGP procedure given in Algorithm 7.2 (page 136). The general idea of the algorithm is to process each position of the motif m, received as parameter, in a greedy fashion. Variable i identifies the motif position being processed. It is initialized with the value 0 (Step 1), the first position of m, and it is incremented in a circular way using

Symbol	Meaning
Σ	IUPAC alphabet
f	input sequences
f_i	i-th input sequence
f_{ij}	j-th position of the i-th input sequence
N	number of input sequences
n_i	length of f_i
k	motif size
S_p	p-th prior (in PSP format)
ℓ	number of priors (it can be zero)
S	$S = \langle S_1, \ldots, S_\ell \rangle$ is the list of all priors
z_{min}	minimum number of motifs expected to be returned by a RISOTTO run
z_{max}	maximum number of motifs expected to be returned by a RISOTTO run
z	number of top motifs post-processed from RISOTTO output
$\mathcal{C}$	the set with the z top motifs to be post-processed by GRISOTTO
m	motif of size k
$m\langle i, \alpha \rangle$	motif m where the i-th position (starting with 0) is replaced by $\alpha \in \Sigma$
ε	empty motif (its BIS score is the minimum possible value: $-\infty$)
P_m	probability distribution given by the PSSM induced by m
$f_i[j \ldots j+k-1]$	k-long segment of the i-th input sequence that starts at position j
$S_p[i, j]$	probability at the j-th position of f_i given by the p-th prior
j_i	annotated position for f_i with maximum BIS score for a motif m
α_p	the weight of the p-th prior
λ	coefficient to balance priors and over-representation contribution

Table 7.1: Definition of terms used in Chapter 7.

Algorithm 7.1 GRISOTTO, Greedy RISOTTO

```
GRISOTTO(DNA sequences $f$, list of priors $S = \langle S_1, \ldots, S_\ell \rangle$)
1. run RISOTTO($k$,$z_{min}$,$z_{max}$);
2. let $r = \varepsilon$ and $\mathcal{C}$ be the list of the first $z$ motifs returned in Step 1;
3. for each motif $m$ in $\mathcal{C}$
4.   let $m$ =GGP($m, f, S$);
5.   if (BIS($r$,$f$,$S$)<BIS($m$,$f$,$S$))
6.     let $r = m$;
7. return $r$;
```

modular arithmetics (Step 9). GPP terminates when k consecutive positions of the motif m being considered can not be improved, according to BIS, and so m remains unchanged for a complete k-round. This information is stored in variable t that counts how many consecutive positions of m have not been modified. Variable t is initialized with 0 (Step 1) and controls the outer cycle (Step 2-9), which terminates when $t = k$. The Boolean flag *changed* is read in the outer cycle (Step 7) to detect whether the i-th position of the motif has been modified inside the body of the inner cycle (Step 6). It is initialized in each run of the outer cycle with $false$ (Step 3). The inner cycle (Step 4-6) tries to improve the BIS score of m by updating its i-th position with each letter $\alpha \in \Sigma$. We denote by $m\langle i, \alpha\rangle$ the motif m where the i-th position of m was replaced by the letter α. Whenever the BIS score of $m\langle i, \alpha\rangle$ is greater than the BIS score of m (Step 5) three variables are updated: (i) motif m is updated to $m\langle i, \alpha\rangle$; (ii) variable t is reset to its initial value, forcing a complete k-round from that point on; and (iii) flag *changed* is turned to $true$. After the inner cycle, in Step 7, we test whether the i-th position of m was not modified by checking the value of the flag *changed*. If that is the case, variable t is incremented (Step 8). Next, in Step 9, variable i is incremented so that the next position of m can be inspected.

We note that the GGP procedure converges since the BIS score is upper-bounded. Next, we derive and present in detail the BIS score.

7.2 Balanced information score

Start by noticing that a motif m of size k written in IUPAC can be easily translated into a PSSM with dimension $4 \times k$ (for details see Appendix C). Moreover, observe that if we had to

Algorithm 7.2 GGP, GRISOTTO greedy procedure

```
GGP(motif $m$, DNA sequences $f$, list of priors $S = \langle S_1, \dots, S_\ell \rangle$)
 1. let $t = 0$ and $i = 0$;
 2. while ($t < k$)
 3.   let $changed = false$;
 4.   for each $\alpha$ in $\Sigma$
 5.     if (BIS($m\langle i, \alpha \rangle$,$f$,$S$)>BIS($m$,$f$,$S$))
 6.       let $m = m\langle i, \alpha \rangle$, $t = 0$ and $changed = true$;
 7.   if (not $changed$)
 8.     let $t = t + 1$;
 9.   let $i = (i + 1) \mod k$;
10. return $m$;
```

guess in which position m occurs in sequence f_i that would be the position j_i that maximizes $P_m(f_i[j_i \dots j_i + k - 1])$ where $P_m(w)$ is the probability of observing the DNA word w by the PSSM induced by m and $f_i[j_i \dots j_i + k - 1]$ is the k-long segment of f_i that starts at position j_i. In other words, such j_i annotates the position in which we believe the motif m occurs in f_i. Henceforward consider that we annotate for each sequence f_i the respective position j_i where m occurs with higher probability (refer to Table 7.1 at page 134).

Following Shannon, the *self-information* of a probabilistic event with probability p is given by $-\log(p)$. If the event is very rare, the self-information is very high. On the other hand, if the event has probability close to 1, observing such event gives us almost no information. So, by assuming that m occurs independently in each sequence of f, the self-information that m occurs in all sequences of f in the annotated positions is given by

$$\sum_{i=1}^{N} -\log(P_m(f_i[j_i \dots j_i + k - 1])). \tag{7.1}$$

Note that the above sum is zero (its minimal value) if the motif m occurs with probability 1 in all annotated positions and, moreover, the sum is not upper-bounded.

If the priors are in PSP format, their information can be easily computed from the annotated sequences. Indeed, the self-information given by the prior S_p of observing the annotated positions j_i, for all $1 \leq i \leq N$, is computed as

$$\sum_{i=1}^{N} -\log(S_p[i, j_i]),$$

where $S_p[i,j]$ is the prior probability stored at the j-th position of the i-th sequence in the S_p PSP file. Having this, it remains to understand how the information from different priors can be combined. Actually, priors come from different sources (Narlikar et al., 2006, 2007; Gordân et al., 2008; Gordân and Hartemink, 2008; Gordân et al., 2010), and some of these sources might have more quality or be more relevant for motif discovery than others. A simple way to heuristically combine prior information is to multiply the contribution of each prior by a constant α_p that measures the belief in the quality/relevance of each prior S_p and consider a balanced sum of all self-informations. In order to keep the resulting value with the same magnitude of each component, we consider a convex combination, that is, $\sum_{p=1}^{\ell} \alpha_p = 1$. Thus, the combined self-information is computed as

$$\sum_{p=1}^{\ell} \left(\alpha_p \sum_{i=1}^{N} - \log(S_p[i, j_i]) \right) . \tag{7.2}$$

Following a similar idea, we balance with a constant $\lambda \in (0,1]$ the self-information given by the occurrence of the motif in (7.1) with the combined self-information given by the priors in (7.2), obtaining in this way the following expression:

$$\lambda \sum_{i=1}^{N} - \log(P_m(f_i[j_i \dots j_i + k - 1])) + (1-\lambda) \sum_{p=1}^{\ell} \left(\alpha_p \sum_{i=1}^{N} - \log(S_p[i, j_i]) \right) =$$
$$- \sum_{i=1}^{N} \left(\lambda \log(P_m(f_i[j_i \dots j_i + k - 1])) + (1-\lambda) \sum_{p=1}^{\ell} \alpha_p \log(S_p[i, j_i]) \right) . \tag{7.3}$$

The closer the above expression is to zero the less (balanced) self-information follows from observing a candidate motif m in the annotated positions of both the DNA sequences and the priors. Indeed, we expect motifs to occur in the annotated positions of both the DNA sequences and the priors with high probability. Therefore, the goal is to find a motif m that minimizes such information. Next, and for the sake of simplification, we drop the minus sign in (7.3), that is, we consider the final scoring criterion, called *balanced information score* (BIS), defined as

$$\mathrm{BIS}(m, f, S) = \sum_{i=1}^{N} \left(\lambda \log(P_m(f_i[j_i \dots j_i + k - 1])) + (1-\lambda) \sum_{p=1}^{\ell} \alpha_p \log(S_p[i, j_i]) \right) , \tag{7.4}$$

and restate our goal to finding a motif m that maximizes (7.4). Note that $\mathrm{BIS}(m, f, S)$ is always non-positive and, therefore, is upper-bounded by 0.

For the BIS score in Equation (7.4) to be well-defined it remains to determine the values of the constants λ and α_p for all $1 \leq p \leq \ell$. Whenever there is no knowledge about the quality of the priors the values of such constants should be uniform, that is, $\lambda = \frac{1}{2}$ and $\alpha_p = \frac{1}{\ell}$ for all $1 \leq p \leq \ell$. Usually, it is possible to refine heuristically these constants by evaluating the usefulness of each prior in well-know domains.

To conclude, we would like to point out that it is not obvious how to translate back the combined information into a combined prior that could be used in an EM or Gibbs sampler-based algorithm. These techniques need that such prior reflects the probability of finding a motif in a certain position of the DNA sequences in order to correctly bias, in each iteration step, the expected log-likelihood of the candidate motif occurring in the positions given by the latent variable. On the other hand, GRISOTTO incorporates prior information in BIS resulting in a theoretical-information scoring criterion that measures the information of observing the candidate motif in the annotated positions of both the DNA sequences and the priors. These annotated positions are computed only once, for each candidate motif, in such a way that the balanced contribution to the BIS score of the DNA sequences and the priors in those positions is maximal. The higher the value of the BIS score, the higher the probability that a candidate motif occurs in the annotated positions of both the DNA sequences and the priors. Therefore, GRISOTTO reports the motif, among all candidate ones, that maximizes the BIS scoring criterion.

7.3 Experimental results

The GRISOTTO algorithm was implemented in Java. Source code and binaries are available at the GRISOTTO webpage.[1]

We start the evaluation of the effectiveness of GRISOTTO by measuring the benefits of using single and combined priors in finding motifs in yeast ChiP-chip data. This data is used as a *gold standard* with several priors available, providing an unbiased experimental assay for motif discovery tools. It contains a human-curated set of 156 motifs known to be present in 156 sequence-sets (one motif per sequence-set). Motif finder tools are asked to report a single motif for each sequence-set, which is then compared with the human-curated one. Human-curated motifs are called throughout this work as *literature motifs*, *known motifs* or even *true*

[1] `http://kdbio.inesc-id.pt/~asmc/software/grisotto.html`

motifs. Details about the data, priors, evaluation methodology, and results can be found in the following ChiP-chip data subsection.

We also provide an additional check on the value of using priors with GRISOTTO from data with different characteristics (a higher eukaryote with sequence data derived from a different technology) by evaluating the performance of GRISOTTO in 13 sequence-sets from mouse ChiP-seq data. Details of this assessment can be found in the ChiP-seq data subsection.

7.3.1 ChiP-chip data

We gauge the performance of GRISOTTO by measuring the benefits of using BIS for finding motifs in 156 sequence-sets experimentally verified to bind different TF's in yeast. These datasets were collected by PRIORITY researchers (Narlikar et al., 2007) and were compiled from the work of Harbison et al. (2009). More precisely, Harbison et al. (2009) profiled the intergenetic binding locations of 203 TF's under various environmental conditions over 6140 yeast intergetecic regions. From these, only intergenetic sequences reported to be bounded with a p-value ≤ 0.001 for some condition were considered by the PRIORITY researchers. Moreover, only sequence-sets with at least size 10 bounded by TF's with a known consensus from the literature were considered, resulting in 156 sequence-sets. Presently, these datasets are being used to benchmark several motif discovery tools (Wang and Stormo, 2003; Sinha et al., 2004; Bailey and Elkan, 1995b; Harbison et al., 2009; Siddharthan et al., 2005; Kellis et al., 2003; Liu et al., 2004; MacIsaac et al., 2006; Bailey et al., 2010; Gordân et al., 2010, 2008; Gordân and Hartemink, 2008; Narlikar et al., 2007) as they provide a reliable and fair test on real data.

Three different PSP's were incorporated in BIS to boost GRISOTTO motif discoverer, namely, priors based on evolutionary conservation (Gordân et al., 2010, 2008), destabilization energy (Gordân and Hartemink, 2008), and nucleosome occupancy (Narlikar et al., 2007). All these priors were devised by PRIORITY researchers and were kindly made available by the authors (personal communication). The popular MEME algorithm was also evaluated with the evolutionary conservation-based prior (Bailey et al., 2010) devised by PRIORITY researchers. Since the sequence-sets and priors used to evaluate GRISOTTO were exactly the ones used in PRIORITY and MEME and, moreover, the criterion used to determine a correct prediction by the algorithms was also the same, we were able to make direct comparisons with their published results. PRIORITY and MEME had already shown that the use of these priors is advantageous when combined with Gibbs sampling and EM techniques. Herein we aim at investigating if the same improvements are also achieved by GRISOTTO. Moreover, we evaluate if combining priors is beneficial.

Following the approach of PRIORITY, we let GRISOTTO look for a single motif of size 8 in each of the 156 yeast sequence-sets, since priors were computed for 8-mers. The results provided by MEME considered a modification of the priors, adapting them for k-mers of different sizes. As a consequence, MEME was able to report accurately a large number of long motifs. Although we acknowledge that MEME's approach improves the capacity to discover motifs, we keep the original priors used in PRIORITY. Moreover, to measure the accuracy of GRISOTTO we used exactly the same metric as the one previously used by the PRIORITY and MEME researches. This metric compares the single motif reported by the discoverer, for each of the 156 yeast sequence-sets, to a literature motif by computing a scaled version of the Euclidean distance between the true motif and the reported one. A more complete explanation of this metric can be found in Appendix C (page 167).

The results of GRISOTTO, as well as those of state-of-the-art motif discoverers, are summarized in Table 7.2 (page 141). The results of motif discoverers were taken from Gordân et al. (2010) and Bailey et al. (2010). All priors used were devised by R. Gordân, A. J. Hartemink and L. Narlikar (Gordân et al., 2008, 2010; Narlikar et al., 2007; Gordân and Hartemink, 2008). Detailed results of GRISOTTO, sequence-set by sequence-set, can be found in the GRISOTTO webpage,[2] while details about the evaluation methodology, including, parameter settings and running times, can be found in Appendixes B–D (page 163). A brief explanation about the priors is given in the following sections.

Evolutionary conservation-based priors

Diverse methods for motif discovery make use of orthologous conservation to assess wether a particular DNA site is conserved across related organisms, and thus more likely to be functional. A comprehensive work along this line was done by PRIORITY researchers (Gordân et al., 2008, 2010), where an orthologous conservation-based prior was devised to improve their Gibbs sampler-based motif discovery method. This prior was built in a discriminative

[2]http://kdbio.inesc-id.pt/~asmc/software/grisotto.html

Algorithm	**Description**	**Successes**	**%**
PhyloCon	Local alignment of conserved regions	19	12%
PhyME	Alignment-based with EM	21	13%
MEME:OOPS	MEME with OOPS model	36	23%
MEME:ZOOPS	MEME with ZOOPS model	39	25%
MEME_c	MEME without conserved bases masked	49	31%
PhyloGibbs	Alignment-based with Gibbs Sampling	54	35%
Kellis *et al.*	Alignment-based	56	36%
CompareProspector	Alignment-based with Gibbs sampling	64	41%
Converge	Alignment-based with EM	68	44%
MEME:OOPS-$\mathcal{DC}$	MEME with OOPS model and $\mathcal{DC}$ priors	73	47%
PRIORITY-$\mathcal{DC}$	Gibbs sampler with $\mathcal{DC}$ priors	77	49%
MEME:ZOOP-$\mathcal{DC}$	MEME with ZOOPS model and $\mathcal{DC}$ priors	81	52%
GRISOTTO-$\mathcal{DC}$	GRISOTTO with $\mathcal{DC}$ priors	**83**	**53%**
PRIORITY-$\mathcal{DE}$	Gibbs sampler with $\mathcal{DE}$ priors	70	45%
GRISOTTO-$\mathcal{DE}$	GRISOTTO with $\mathcal{DE}$ priors	**80**	**51%**
PRIORITY-$\mathcal{DN}$	Gibbs sampler with $\mathcal{DN}$ priors	70	45%
GRISOTTO-$\mathcal{DN}$	GRISOTTO with $\mathcal{DN}$ priors	**77**	**49%**
GRISOTTO-$\mathcal{CDP}$	GRISOTTO with combined priors	**93**	**60%**

Table 7.2: Comparison of GRISOTTO with state-of-the-art methods over ChiP-chip data.

way by taking into account not only sequence-sets that were bounded by some profiled TF (the positive set) but also sequence-sets that were not bounded by the same TF (the negative set). In this way the prior reflects not only the probability that a W-mer at a certain position is conserved but of all the conserved occurrences of this W-mer what fraction occurs in the bound sequence-set. Conserved occurrences are found by searching if a W-mer in a reference sequence also occurs in most of its orthologous ones regardless of its orientation or specific position. For this particular case, the evolutionary conservation-based prior was used for each intergenetic region in *S. cerevisiae* and it used the orthologous sequences from six related organisms, namely, *S. paradoxus*, *S. mikatae*, *S. kudriavzevii*, *S. bayanus*, *S. castelli* and *S. kluyveri*. The prior was named *discriminative conservation-based prior* ($\mathcal{DC}$) and was made available, in a PSP format, at the PRIORITY webpage.

Herein, we gauge the performance of GRISOTTO when this exact $\mathcal{DC}$ prior is incorporated into the BIS scoring criterion. Results comparing GRISOTTO-$\mathcal{DC}$ with PRIORITY-$\mathcal{DC}$ (Gordân et al., 2010), MEME-$\mathcal{DC}$ (Bailey et al., 2010), and other state-of-the-art algorithms, can be found in Table 7.2 (page 141). Results show that GRISOTTO-$\mathcal{DC}$ correctly predicted 83 motifs out of the 156 experiments, whereas PRIORITY-$\mathcal{DC}$ found 77 and MEME:ZOOP-$\mathcal{DC}$ 81. We conclude that GRISOTTO performed at least as well as PRIORITY and MEME:ZOOP when the same $\mathcal{DC}$ PSP was used. A closer inspection of detailed results of GRISOTTO, available at the GRISOTTO webpage, reveals that GRISOTTO-$\mathcal{DC}$ found 15 motifs that PRIORITY-$\mathcal{DC}$ did not, while PRIORITY-$\mathcal{DC}$ found only 10 motifs that GRISOTTO-$\mathcal{DC}$ did not.

Destabilization energy-based priors

Information about DNA double-helical stability has also been collected into a PSP to boost the PRIORITY Gibbs sampler-based algorithm (Gordân and Hartemink, 2008). The rational for the information contained in this prior is based in the fact that, in general, the energy needed to destabilize the DNA double helix is higher at TFBS's than at random DNA sites. The PSP resulting from this effort was built in a discriminative way, just as for the $\mathcal{DC}$ prior, and was named *discriminative energy-based prior* ($\mathcal{DE}$).

We evaluated the $\mathcal{DE}$ prior within GRISOTTO. Results comparing GRISOTTO-$\mathcal{DE}$ with PRIORITY-$\mathcal{DE}$ (Gordân and Hartemink, 2008), and other state-of-the-art algorithms, can be found in Table 7.2 (page 141). This table shows that GRISOTTO-$\mathcal{DE}$ correctly predicted 80 motifs out of the 156 experiments, whereas PRIORITY-$\mathcal{DE}$ found only 70. We conclude that GRISOTTO performed quite well when the $\mathcal{DE}$ prior was used, with an improvement of 14% over correct predictions relatively to PRIORITY, raising the overall proportion of successful predictions in 6% (from 45% to 51%). As before, we made a closer examination of the detailed results included in an additional file at the GRISOTTO webpage which revealed that GRISOTTO-$\mathcal{DE}$ found 19 motifs that PRIORITY-$\mathcal{DE}$ did not, whereas PRIORITY-$\mathcal{DE}$ found only 9 motifs that GRISOTTO-$\mathcal{DE}$ did not.

Nucleosome occupancy-based priors

Nucleosome occupancy has also been used in motif discovery. The rationale for this approach is that Eukaryotic genomes are packaged into nucleosomes along chromatin affecting sequence accessibility. There are three main works in the literature to predict genome-wide organization of nucleosomes in *Saccharomyces cerevisiae* (Lee et al., 2004; Yuan et al., 2005; Segal et al., 2006). Taking into account the work of Segal et al. (2006) the PRIORITY researchers (Narlikar et al., 2007) devised an informative prior based on a discriminative view of nucleosome occupancy. The prior was named *discriminative nucleosome-based prior* ($\mathcal{DN}$).

GRISOTTO was evaluated with the $\mathcal{DN}$ prior incorporated in the BIS score. Results comparing GRISOTTO-$\mathcal{DN}$ with PRIORITY-$\mathcal{DN}$, and other state-of-the-art algorithms, can be found in Table 7.2 (page 141). This table shows that GRISOTTO-$\mathcal{DN}$ correctly predicted 77 motifs out of the 156 experiments, while PRIORITY-$\mathcal{DC}$ found 70. We conclude that GRISOTTO outperformed PRIORITY when $\mathcal{DN}$ prior was used, with an improvement of 10% over correct predictions. A closer investigation of detailed results in an additional file at the GRISOTTO webpage unravels that GRISOTTO-$\mathcal{DN}$ found 13 motifs that PRIORITY-$\mathcal{DN}$ did not, whereas PRIORITY-$\mathcal{DN}$ found 6 motifs that GRISOTTO-$\mathcal{DN}$ did not.

Combining priors

Despite considerable effort to date in developing new potential priors to boost motif discoverers, PSP's from different sources have not yet been combined. Actually, although having some degree of redundancy, because, for instance, the positioning of nucleosomes may be correlated with DNA double helix stability, it is easy to conclude by a closer inspection of the detailed results in an additional file at the GRISOTTO webpage that different PSP's still report a considerable number of disjoint motifs (refer to Appendix D.3 for further details). As a matter of fact, PRIORITY researchers have already noticed this fact (Gordân and Hartemink, 2008). However, it is not a trivial task determining how to translate the BIS combined information into a PSP that can be used in EM or Gibbs sampler-based algorithms.

In order to gauge the potential of combined priors, we incorporated in the BIS score the three $\mathcal{DC}$, $\mathcal{DE}$ and $\mathcal{DN}$ priors. We call the final prior *combined discriminative prior* ($\mathcal{CDP}$). Results show that GRISOTTO-$\mathcal{CDP}$ is the more accurate motif discoverer for the 156 sequence-sets being evaluated. It correctly predicted 93 motifs, while GRISOTTO-$\mathcal{DC}$ found 83, GRISOTTO-$\mathcal{DE}$ 80 and GRISOTTO-$\mathcal{DN}$ 77. In this way GRISOTTO-$\mathcal{CDP}$ accomplished an improvement of at least 12% over correct predictions, when compared with GRISOTTO variants considering the priors individually. This raises the overall proportion of successful predictions in 7%, on top of the improvements already attained in the previous sections, over these 156 yeast sequence-sets. Moreover, when comparing GRISOTTO-$\mathcal{CDP}$ with state-of-the-art motif discoverers (refer to Table 7.2, page 141), the final proportion of successful predictions was raised to 60%, while the best known previous value, to our knowledge, was 51% attained by MEME-$\mathcal{DC}$ (Bailey et al., 2010). This leads us to conclude that combining priors from different sources is even more beneficial than considering them separately.

7.3.2 ChiP-seq data

Herein, we measure the accuracy of GRISOTTO in motif discovery on 13 mouse ChiP-seq data. This data was gathered by Chen et al. (2008) where whole-genome binding sites of 13 sequence-specific TF's (Nanog, Oct4, STAT3, Smad1, Sox2, Zfx, c-Myc, n-Myc, Klf4, Essrb, Tcfcp2l, E2f1, and CTCF) were profiled in mouse ES cells using the ChiP-seq approach. Sequences of ±100bp size from the top 500 binding peaks were selected for each factor, repeats were masked, and the Weeder (Pavesi et al., 2004b) tool was used to find overrepresented sequences unravelling 12 of the 13 factors (excluding E2f1).

We assess the quality of GRISOTTO in discovering motifs from mouse ChiP-seq data with two priors. First, an orthologous conservation-based PSP was used as information for higher organisms is now available. Indeed, there are already such PSP's for yeast, fly, mouse and even human (Bailey et al., 2010; Gordân et al., 2010, 2008). Second, a binding peak-based PSP was tried as ChiP-seq assays provide an intrinsic positional prior that can be computed from base-specific coverage profiles. This prior has recently been employed in motif discoverers (Kulakovskiy et al., 2010; Hu et al., 2010) with success.

As for ChiP-chip data, we let GRISOTTO find for a single motif of size 8, since priors were computed for 8-mers. However, as human-curated motifs are not available for this ChiP-seq data, we made only a resemblance, based on a 6-window match, between the motifs reported by GRISOTTO with those outputted by Chen et al. (2008) and MEME (Bailey et al., 2010) for the same data.

Evolutionary conservation-based priors

Orthologous conservation-based priors for mouse ChiP-seq data were obtained by MEME researchers (Bailey et al., 2010) following a similar methodology as PRIORITY-$\mathcal{DC}$ for the yeast ChiP-chip data ones. As before, this new mouse prior received the shorthand name $\mathcal{DC}$. We incorporated the $\mathcal{DC}$ prior into the BIS score and ran GRISOTTO. In Table 7.3 (page 146), motifs reported by Chen *et al.* and MEME-$\mathcal{DC}$ are shown along side motifs found by GRISOTTO-$\mathcal{DC}$ for the 13 mouse sequence-sets. Recall that Chen *et al.* only reported 12 out of the 13 motifs, excluding the E2f1 motif, so in this case the TRANSFAC (Matys et al., 2006) motif is shown instead. MEME-$\mathcal{DC}$ and GRISOTTO-$\mathcal{DC}$ were able to retrieve all motifs. Moreover, the number of sequences of these sequence-sets vary from 1038 to 38238 and, due to efficiency issues, MEME-$\mathcal{DC}$ was only able to run over 100 sequences randomly chosen from each sequence-set. GRISOTTO-$\mathcal{DC}$ was able to use all of them taking only 1-4 minutes, per sequence-set, to report a motif.

Because sequences-sets are very large, some of the reported motifs became highly degenerated. Actually, only 6 out of the 13 motifs seem to be highly conserved, namely, CTCF, Esrrb, Klf4, n-Myc, Tcfc and c-Myc. For these, even allowing for IUPAC symbols during the greedy search results in highly conserved motifs. Therefore, for this data, we searched for IUPAC strings that allow only two positions to have degenerate IUPAC symbols.

By a closer inspection of Table 7.3 we conclude that motifs reported by GRISOTTO-$\mathcal{DC}$ are strongly similar to the ones reported by Chen *et al.* and MEME-$\mathcal{DC}$. Have in mind that GRISOTTO outputs an IUPAC, and not a PSSM, but we used, in a 6-window size, the same color scheme as PSSM's to make the resemblance with reported motifs easier.

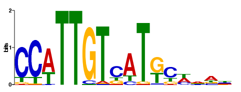
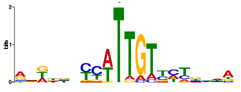
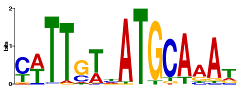
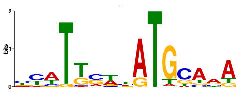
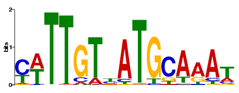
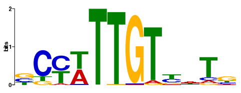
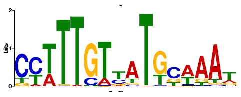
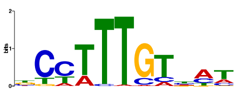
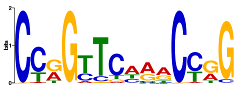
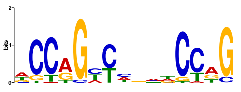
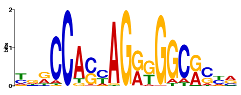
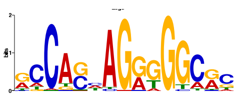
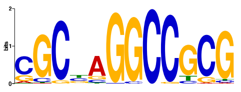
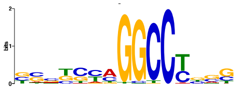
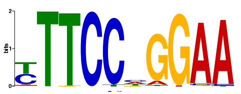
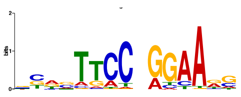
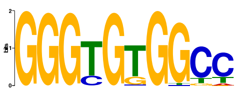
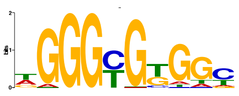
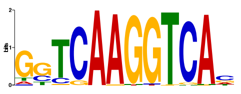
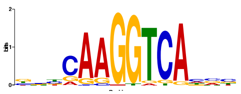
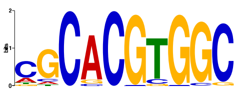
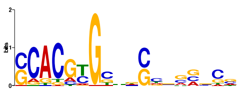
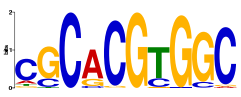
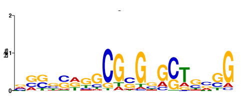
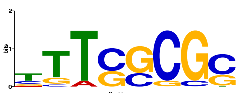
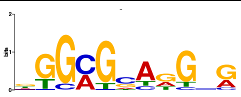
TF	Chen *et al.* motif	MEME-$\mathcal{DC}$ motif	GRISOTTO-$\mathcal{DC}$ motif
Nanog			CC$^{A}_{T}$TTGT$^{C}_{T}$
Oct4			$^{GA}_{CT}$TATGCA
Sox2			CC$^{A}_{T}$TTGT$^{C}_{T}$
Smad1			$^{A}_{T}$TGCC$^{A}_{C}$ATT
Tcfcp2l1			AACCAG$^{CC}_{TT}$
CTCF			AG$^{G}_{A}$GGGC$^{G}_{A}$
Zfx			C$^{C}_{T}$AGGCC$^{C}_{T}$
STAT3			TCC$^{G}_{T}$GGC$^{A}_{A}$
Klf43			GGG$^{C}_{T}$G$^{G}_{T}$GG
Esrrb			$^{G}_{C}$AAGGTCA
c-Myc			$^{G}_{A}$CACG$^{C}_{T}$GG
n-Myc			CACGTG$^{GC}_{CT}$
TF	**TRANSFAC motif**	**MEME-$\mathcal{DC}$ motif**	**GRISOTTO-$\mathcal{DC}$ motif**
E2f1			CTGC$^{CG}_{TT}$CC

Table 7.3: Comparison of GRISOTTO-$\mathcal{DC}$ with Chen *et al.* and MEME-$\mathcal{DC}$ over ChiP-seq data.

Binding peak-based priors

Hu et al. (2010) devised a prior using coverage profile information provided by the ChiP-seq approach. This grounds in the belief that motifs are tightly packed near the *peak summit* – the location inside each peak with the highest sequence coverage depth. As a result, prior probabilities were set to be proportional to a discretized Student's t-distribution with 3 degrees of freedom and rescaled such that they form a step function with a fixed 25bp step-size. The prior probabilities are symmetric and centered at the peak summits. As such prior is intrinsically a positional one we built a PSP resuming the described probabilities for the 13 mouse ChiP-seq data and ran GRISOTTO.

Our results show that direct use of binding peak-based priors does not help GRISOTTO much. Actually, the motifs reported by this prior were exactly the same as using the uniform one for which any position in the DNA is likely to contain a motif. Moreover, when combined with the $\mathcal{DC}$ prior GRISOTTO reported precisely the same motifs as $\mathcal{DC}$ prior alone. These findings suggest that GRISOTTO is unable to retrieve any useful information from the binding peak-based prior. We attributed this to the fact that part of the information contained in the binding peak-based prior is already encoded in the BIS score. Indeed, peak summits indicate an overrepresentation of a motif in a certain locus. Such overrepresentation is already weighted in the BIS score (recall Equation (7.1) and (7.4) in page 136–137). Notwithstanding, it seems reasonable that for short sequences of 200bp (namely, $\pm$100bp around peak summits) the coverage-based prior has no real impact on motif discovery. For longer sequences, the effective resolution of the peak summits seems to provide useful information (see Hu et al., 2010; Kulakovskiy et al., 2010).

7.4 Discussion

Wasserman and Sandelin (2004) noticed that the discovery of TFBS's from a nucleotide sequence alone suffers from impractical high false positive rates. This was termed the *futility theorem* as nearly every predicted TFBS has no function *in vivo*. This problem has been studied and addressed by taking into consideration information in and beyond the TFBS's, such as orthologous conservation (Gordân et al., 2010; Bailey et al., 2010), nucleosome positioning (Narlikar et al., 2007; Daenen et al., 2008), DNA duplex stability (Gordân et al., 2008) and coverage profiles obtained from ChiP-seq assays (Kulakovskiy et al., 2010; Hu et al., 2010).

Following this line of research we have verified in the present study that post-processing the output of RISOTTO with prior knowledge from different sources is beneficial for motif discovery. RISOTTO is a consensus-based method that enumerates exhaustively all motifs by collecting their occurrences, up to a fixed *Hamming distance*[3], from input sequences. As a result, a set of overrepresented motifs is reported and then ordered by their biological relevance according to a statistical significance test (Marsan and Sagot, 2000; Carvalho et al., 2006; Pisanti et al., 2006). This ordered list is retrieved in a classical way from the nucleotide sequence alone and, as previously mentioned, it is of particular importance to introduce a bias from available priors. Following this goal, we noticed that the top 10 motifs from the RISOTTO ordered list could be greedily modified to have a good BIS score. The greedy procedure would modify these motifs introducing some noise allowed by the prior and up-weighting weak motifs that were masked during the combinatorial and/or statistical significance test. Certainly, we would not expect RISOTTO, or any other combinatorial algorithm, to report completely outlandish motifs, as motif discovery problem is indeed a combinatorial problem that accounts for overrepresentation of a string in a set of DNA sequences. However, prior information provides valuable guidance on how to describe a motif that goes beyond neighborhoods (defined by the Hamming distance or any similar distance) of the consensus sequence. GRISOTTO incorporates such information in the BIS score providing in this way a broader definition of overrepresentation of a motif in the input sequences.

Currently, a significant point of discussion is related with the use of prior information as a post-processing step of RISOTTO, and not within the RISOTTO procedure itself. For the sake of simplicity, consider we are looking for motifs of a fixed size k. Combinatorial algorithms take into consideration overrepresentation of motifs to extract them. This extraction is exhaustive, by iteratively extending candidate strings of size $1 \ldots k-1$, letter by letter of the DNA alphabet, and checking in each step if the extended string is still overrepresented in the sequence-set. Usually, complex data structures, such as suffix-trees, are employed to extend the candidate string. Whenever an extension fails to be overrepresented in the input sequences that extension is disregarded and another one is attempted. Only extensions that

[3]The Hamming distance between two string measures the minimum number of substitutions required to change one string into the other.

reach the size k are reported.

Conversely, prior information only asserts if a sub-sequence of a fixed size W in a certain position of the DNA sequences is likely to be a motif. It is not straightforward to use prior information in combinatorial algorithms because they would need to know if a sub-string of size $1 \ldots k-1$ is likely to be a motif. However, in one hand, it is space-wise unfeasible to have priors for multiple values of W. On the other hand, priors for small or large values of W have no information whatsoever, as either they are very common (occur in all input sequences) or very rare (occur only once or never). Our work, as well as state-of-the-art ones (Gordân et al., 2008, 2010; Narlikar et al., 2007; Gordân and Hartemink, 2008; Bailey et al., 2010), have shown that an efficient and effective solution is to consider $W = k = 8$.

Besides this discussion, there are two obvious advantages of using prior information at a post-processing step. First, the greedy-search procedure is independent from the starting points provided by the combinatorial algorithm, allowing any method to be employed; for instance, Weeder (Pavesi et al., 2004b), SMILE (Marsan and Sagot, 2000), RISO (Carvalho et al., 2006), RISOTTO, etc. Another advantage is that while new priors are devised, we do not need to re-compute previous starting points, being sufficient to run the greedy-search procedure of the GRISOTTO algorithm.

In closing, we stress that the BIS score was used throughout the experiments with sequence-sets known to be bound by a TF. Therefore, it was only used to discover the positions of each sequence-set where the motif occurs. Another possible application of the BIS score would be to detect the fraction of sequences that are likely to have site predictions. There are two possible ways to adapt GRISOTTO to this new problem: (i) derive a threshold of the BIS score contribution of a sequence above which the sequence is likely to have site predictions; (ii) incorporate an input parameter in the GRISOTTO greedy procedure, usually called *quorum*, that amounts for the fraction of sequences that have binding site predictions. None of these approaches seems straightforward and are out of the scope of this work, hence they were left as a future research topic.

Part IV

Conclusions and future work

Conclusions

Herein we draw some conclusions concerning the topics of this thesis: motif representation and discovery.

Motif representation

Motifs have been represented in a variety of ways (refer to Section 2.2, page 12). In this thesis we used Bayesian networks and multinets to model transcription factor binding sites (TFBS) and discriminate them from the background DNA sequences. Since distinguishing motifs from the background is chiefly a classification task, we proposed a new discriminative scoring criterion, called factorized conditional log-likelihood ($\hat{\text{f}}$CLL), for learning augmented naive Bayes networks, based on an approximation of conditional log-likelihood (CLL). The new criterion is decomposable, score-equivalent, and allows efficient estimation of both structure and parameters. In this way, we addressed an important problem concerning discriminative learning of Bayesian networks classifiers (BNC) posed by Friedman et al. (1997) in a very efficient way. Actually, the computational complexity of the proposed method is of the same order as the traditional log-likelihood criterion.

The merits of the new scoring criterion were evaluated and compared to those of common state-of-the-art classifiers, on a large suite of benchmark datasets from the UCI repository. Optimal $\hat{\text{f}}$CLL-scored tree augmented naive (TAN) Bayes classifiers (Friedman et al., 1997), as well as somewhat more general structures, performed significantly better than generatively-trained Bayesian network classifiers, as well as C4.5, nearest neighbor, and logistic regression classifiers. Moreover, $\hat{\text{f}}$CLL-optimal classifiers performed better, although the difference is not statistically significant, than those where the Bayesian network parameters were optimized using a gradient descent method to find maximum-CLL parameters, namely, extended logistic regression (ELR) proposed by Greiner et al. (2005), as well as support vector machines (with linear, polynomial and RBF kernels). In comparison to the latter methods, our $\hat{\text{f}}$CLL-based approach is considerably more efficient in terms of computational cost, being 2 to 3 orders of magnitude faster, in 25 benchmark datasets from the UCI machine learning, with minimum memory requirements.

Learning unrestricted BNC's is known to be a NP-complete problem. For this reason, BNC's can only be learned efficiently for restricted structures such as TAN's. However, TAN's are not able to capture important dependencies among binding site positions. To overcome this limitation, we proposed a new heuristic that improves an optimal TAN classifier by adding important dependencies and removing irrelevant ones. This process is guided by the total order induced by the breath-first search (BFS) over the optimal TAN. Given that the resulting BNC is consistent with the total order of the optimal TAN, it was called a consistent κ-graph (CκG). The search space of the proposed heuristic is more general than trees and intersect, but is not contained in, polytrees. Moreover, we show that an optimal CκG can be found in polynomial time, while augmenting the search space exponentially, in the number of nodes/attributes, relatively to trees. The CκG learning algorithm can be applied to any decomposable score. We show that the score of the optimal CκG is always greater than or equal to the score of the optimal TAN and naive Bayes classifiers.

Finally, since there is no reason to assume that a unique CκG model is suitable to represent the promoter regions of co-regulated genes, the background, and at the same time, a motif within such promoter region, the foreground, we proposed to use two-component mixtures of CκG's as promoter/motif models. Indeed, there seems to be two separate underlying regimes, so instead we model the background with a CκG Bayesian network model and the foreground with another CκG Bayesian network model. In this case, the resulting model is a two-component mixture, also called a multinet, of CκG models. Taking this into account, we extended the $\hat{\text{f}}$CLL scoring criterion to mixtures of Bayesian networks. To access the quality of the proposed model, we compared CκG multinets against other Bayesian network models using 89 sequence-sets of TFBS's retrieved from the TRANSFAC database (Wingender et al., 2001). We concluded that our approach outperformed other methods with considerable statistical significance.

Motif discovery

We proposed RISOTTO, a new algorithm for motif discovery, improving the performance of RISO (Carvalho, Freitas, Oliveira, and Sagot, 2006). The improvement consists in storing information concerning maximal extensibility of factors in order to avoid trying to extend hopeless candidate motifs. Experimental results show that the improvement works for large motifs, achieving an improvement of 40% over the computational time of RISO. In terms of space, a trade off between memory allocation/management and maximal extensibility gain was made, leading to at most 1.3MB of memory cost for storing the extensibility information, which is negligible for the current computational power. We also performed an average case analysis of the amount of saved extensions achieved by RISOTTO over RISO. This theoretical analysis was confirmed experimentally.

Unfortunately, combinatorial methods, such as RISO or RISOTTO, tend to output a large number of putative motifs, making hard to elicit which ones have function *in vivo*. Moreover, they are know to face problems in detecting weak motifs. To overcome these issues we introduced the GRISOTTO algorithm, that post-processes in a greedy-fashion the output of RISOTTO taking into account prior information available about the domain. In practice, this prior information introduces some extra knowledge taken from the literature, or computed from the sequences, that will help in characterizing motifs. The algorithm is flexible enough to combine several priors from different sources. Each prior is given a weight reflecting the confidence on the information contained in it and its relevance for motif discovery. In this way, priors can be introduced at will giving rise to a scoring criterion based on the convex closure of the information given by each prior. We called this scoring criterion balanced information score (BIS).

Prior information has previously been shown to be beneficial when used with EM and Gibbs sampler-based motif discoverers. We have shown in this thesis that they can also be of great benefit to boost combinatorial algorithms such as RISOTTO. We emphasize that our goal is not to introduce new priors, but to show that priors can also be advantageous to assist and improve the output of combinatorial algorithms such as RISOTTO. Moreover, we have shown that combining priors is very promising in further extending the power of motif discovery algorithms.

We gauge the effect of adding prior information to GRISOTTO over 156 well-studied sequence-sets from yeast ChiP-chip experiments. For each sequence-set, motif discoverers were asked to report a single position specific scoring matrix (PSSM) that was then compared with the known PSSM for the transcription factor pulled down in the ChiP-chip experiment. Prior information from different sources was used, including, orthologous conservation, nucleosome occupancy, and destabilization energy. The use of exactly the same priors in EM and Gibbs sampler-based motif discoverers, namely, MEME (Bailey et al., 2010) and PRIORITY (Narlikar et al., 2006, 2007; Gordân et al., 2008; Gordân and Hartemink, 2008; Gordân et al., 2010), respectively, has been shown to dramatically improve their performance. In this thesis, we show that this boost can be also achieved by GRISOTTO that performed at least as well as PRIORITY and MEME when each prior was considered individually. The great advantage of GRISOTTO was accomplished by the combination of priors. Indeed, when GRISOTTO integrated the three mentioned priors in a convex combination of their information it achieved an improvement of about 15% over correct predictions relatively to the best motif discoverer (MEME-$\mathcal{DC}$ proposed by Bailey et al., 2010), at our present knowledge, for exactly the same experiments. The final proportion of successful predictions is now at 60%, attained with 93 correct predictions from GRISOTTO-$\mathcal{CDP}$ (with only 81 correct predictions of MEME-$\mathcal{DC}$) out of the 156 experiments.

Finally, we also confirmed the benefit of using GRISOTTO with 13 sequence-sets from a higher eukaryote ChiP-seq data, namely, the mouse. In this assessment two priors were used, including, orthologous conservation and base coverage profiles obtained from the ChiP-seq assays. We concluded that, as for ChiP-chip data, the orthologous conservation-based prior was of great convenience, being able to unravel 13 motifs strongly similar to the ones reported by other tools and found in the TRANSFAC database. In respect to the coverage-based prior, their direct use as a positional prior was not favorable, having been comparable to the uniform prior. We believe this is due to the fact that the BIS score already accounts for overrepresentation in the input sequences which we suspect mimics the information contained in this new prior, turning the prior redundant.

Future work

Discriminative learning of Bayesian networks and multinets

Directions for future work concerning aCLL and $\hat{\text{f}}$CLL scoring criteria include studying in detail the asymptotic behavior of $\hat{\text{f}}$CLL for TAN and more general models, combining our intermediate approximation, aCLL, with discriminative parameter estimation (ELR) and extending and studying aCLL and $\hat{\text{f}}$CLL to mixture models. Finally, adapting the devised scores to applications in data clustering should yield interesting results.

Studying other applications for the discriminative learning of multinets

Although discriminative learning of mixtures of CκG exhibit a great promise in modeling TFBS's its usage is not restricted to this single application. Actually, there are plenty of applications where discriminative mixtures of CκG can be used providing good probabilistic models for classification tasks. One of these tasks is medical diagnosis. As a matter of fact, in an attempt to further exploit the value of mixtures of CκG we did a preliminary evaluation over the UCI machine learning datasets, the same 25 datasets used to evaluate $\hat{\text{f}}$CLL in Section 4.4 (page 82). Results showed that Algorithm 5.3 (page 108) was very effective when learning two-component mixtures of C2G Bayesian network models, specially in diagnosis datasets (e.g., breast, diabetes and heart). These results are reasonable and arguably related to the fact that CκG is a Bayesian network that allows v-structures and these structures are known to be good in diagnosis, on top of the fact that such networks are learned in a discriminative manner. Indeed, v-structures represent the so-called *induced dependencies* where totally unrelated propositions became relevant to each other when new facts are learned. In the case of diagnosis the new facts are the symptoms. This topic is, however, far from the thesis subjects so we left it to follow-up work.

Devising new priors for motif discovery

Another promising application of discriminative learning of mixtures of CκG's is in devising new PSP's for posterior use in motif discovery tools. This discriminative model enable us to learn a set of relevant features that characterize TFBS's, including, not only dependencies among TFBS positions (as it was done in Chapter 5), but also some extra features that may be heuristically calculated, based on sequence data, or taken from external annotation. Such extra features may include, architecture of the regulatory region, presence of repeats, an evolutionary score-based feature, GC-content, melting temperature, nucleosome occupancy, reverse complementarity and conservation symmetry.

Additional follow-up work directly related with this topic includes studying the effect of the size of the sub-sequence used to build the prior in motif discovery and understanding how to devise better priors for motifs with spacers. Concerning the latter, we verified that existing priors were not able to describe much more than half of the motifs with spacers present in the sequence-sets.

Part V

Appendixes

Appendix A

Alternative justification for Assumption 1

Observe that in the case at hand we have some information about U_t and V_t, namely, the number of times, say N_{U_t} and N_{V_t}, respectively, that U_t and V_t occur in the dataset T. Moreover, we also have the number of times, say $N_{R_t} = N - (N_{U_t} + N_{V_t})$, that R_t is found in T. So, under this knowledge, we have that

$$(U_t, V_t) \sim \text{Dirichlet}(N_{U_t} + 1, N_{V_t} + 1, N_{R_t} + 1). \tag{A.1}$$

Furthermore, we know that N_{U_t} and N_{V_t} are, in general, a couple (or more) orders of magnitude smaller than N_{R_t}. Due to this fact, most of all probability mass of (A.1) is found in the square $[0, p] \times [0, p]$ for some small p.

Take as an example the (typical) case where $N_{U_t} = 1$, $N_{V_t} = 0$, $N = 500$ and

$$p = E[U_t] + \sqrt{Var[U_t]} \approx E[V_t] + \sqrt{Var[V_t]},$$

and compare the cumulative distribution of Uniform($[0, p] \times [0, p]$) with the cumulative distribution of Dirichlet($N_{U_t}+1, N_{V_t}+1, N_{R_t}+1$) in the supplementary material webpage. Actually, the cumulative distribution Dirichlet($N_{U_t}+1, N_{V_t}+1, N_{R_t}+1$) when N_{R_t} is much larger than N_{U_t} and N_{V_t} is close to the Uniform($[0, p] \times [0, p]$) for some small p, and so Assumption 1 follows naturally.

Concerning independence, and by assuming that the distribution of (U_t, V_t) is given by

(A.1), it results from the neutrality property of the Dirichlet distribution that

$$V_t \perp\!\!\!\perp \frac{U_t}{1 - V_t}.$$

Since V_t is very small we have

$$V_t \perp\!\!\!\perp \frac{U_t}{1 - V_t} \approx U_t.$$

Therefore, it is reasonable to assume that U_t and V_t are independent.

Finally, note that (A.1) fails to give us information for establishing a reasonable distribution over a general U_t and V_t, at least one distribution that is suitable for further analysis (since it depends on the dataset T, and moreover, on its t-th instance). This is the reason why we use an uniform distribution for (U_t, V_t) in Assumption 1 instead of (A.1).

Appendix B

Feeding GRISOTTO with good initial starting points

Herein we describe the call to RISOTTO algorithm found in Step 1 of the Algorithm 7.1 (page 135). This call tries to tune the RISOTTO input, presented in Section B.1, in order to obtain good initial starting points to be processed by GRISOTTO. In Section B.2, a description of the core idea and the pseudocode of the tuning procedure is provided.

B.1 RISOTTO input

RISOTTO(Pisanti, Carvalho, Marsan, and Sagot, 2006) is a consensus-based combinatorial algorithm that finds all motifs of size k by collecting their occurrences, at a given distance, from a set of N co-regulated DNA promoter sequences. The motif occurrences should be at Hamming distance at most e from the motif consensus string, where e is called the *number of mismatches*. Moreover, the motifs need not to occur in every input sequence but in at least q% of the N sequences, where q is called the *quorum*. Furthermore, the quorum q must cover at least two input sequences. After reporting all consensus, RISOTTO orders them by statistical significance using a program from the SMILE package (Marsan and Sagot, 2000). To sum up, the inputs of RISOTTO algorithm are:

- set of DNA sequences $f = (f_i)_{i=1...N}$;
- quorum percentage $q \in \{1, \ldots, 100\}$;

- number of mismatches e;
- motif size k.

The source code and executables of RISOTTO are available in its webpage at

`http://kdbio.inesc-id.pt/ asmc/software/riso.html`.

B.2 GRISOTTO subroutine calling RISOTTO

GRISOTTO first step (refer to Step 1 of GRISOTTO algorithm in page 135) calls RISOTTO in order to be provided with good starting points for the greedy procedure. Herein we discuss how GRISOTTO tunes RISOTTO parameters to achieve this.

Given that GRISOTTO capitalizes in the PSP's used and devised by PRIORITY researches (Narlikar et al., 2006, 2007; Gordân et al., 2008; Gordân and Hartemink, 2008; Gordân et al., 2010) and that those PSP's were devised for 8-mers, the RISOTTO algorithm will always be run with $k = 8$. Clearly, if different PSP's are considered, different values for k should be considered as well. Moreover, since the list of sequences f is fixed, the only parameters that GRISOTTO needs to tune in RISOTTO runs are q and e. GRISOTTO uses 5 different variables to tune q and e:

- q-max, the maximum quorum acceptable, its default value is 100;
- q-min, the minimum quorum acceptable, its default value is 5;
- q-step, the decrement step-size to modify the RISOTTO quorum, its default value is 5;
- nb-max, the maximum number of motifs reported by RISOTTO, its default value is 80;
- nb-min, the minimum number of motifs reported by RISOTTO, its default value is 50.

The goal is to find the largest quorum q, with q-min $\leq q \leq$ q-max, and minimal error e such that the number of motifs reported by RISOTTO is between nb-min and nb-max. Algorithm B.1 describes this procedure and prefers decrementing the quorum to augmenting the error. Recall that RISOTTO needs four parameters and so it is formally called with

RISOTTO(DNA sequences f, quorum q, motif size k, mismatches e).

Algorithm B.1 RunRISOTTO, RISOTTO parameter tuning

```
RunRISOTTO(mismatches e)
 1. for q :=q-max to q-min with step-size (−q-step) do// ranging q from q-max to q-min decrementing q-step
 2.   nb := length(RISOTTO(f,q,k,e)); // nb stores the number of motifs return by RISOTTO
 3.   if (nb>nb-max) then
 4.     if (q≥q-max) then return (q,e); // there is no way to reduce the number of motifs
 5.     else break; // get out of the for loop and refine at most 3 times the quorum q in Step 9
 6.   else if (nb≤nb-max && nb≥nb-min) then return (q,e); // found the correct values for q and e
 7.   else // the case when nb<nb-min
 8.     if (q≤q-min) then return RunRISOTTO (e+1); // recursive call, consider more mismatches to get more motifs
 9. for i := 1 to 3 do// refine the quorum at most 3 times
10.   if (nb>nb-max) then q + := q-step/$2^i$;
11.   else if (nb<nb-min)) then q − := q-step/$2^i$;
12.   else return (q,e);
13.   nb := length(RISOTTO(f,q,k,e));
14. return (q,e);
```

Algorithm B.1 is self-explanatory, we just note that RunRISOTTO is initially called with zero number of mismatches, that is, the algorithm is called as RunRISOTTO(0). It returns the pair (q,e) with largest quorum and minimum number of mismatches such that the number of motifs reported by RISOTTO is between nb-min and nb-max. In general, this is not always possible. Actually, there are two distinct situations where this can happen. First, when the algorithm reaches the Step 5. Second, when Step 8 fails and q−q-step<q-min which makes the guard of the for-loop in Step 1 also to fail. In these two cases there is a jump to Step 9 where the quorum is refined, at most 3 times, in order to achieve the expected number of motifs (Step 9-14). If with this refinement the number of motifs still remains larger than nb-max, or smaller than nb-min, then the run that produced the number of outputs closer to the expected ones is chosen. In practice, for the experiments considered, only a few sequence-sets (1 or 2 out of 156) failed to report a number of motifs between nb-min and nb-max.

Finally, we note that for the sequence-sets considered in this work this tuning was achieved in much less than 1 second per sequence-set.

Appendix C

Inter-motif distance

To assess the accuracy of GRISOTTO presented in Chapter 7 we used a scaled version of the Euclidean distance between PSSM's, exactly the same metric used in PRIORITY and MEME works (Narlikar et al., 2007; Gordân et al., 2008; Gordân and Hartemink, 2008; Gordân et al., 2010; Bailey et al., 2010). Since both literature motif and top scoring motif reported by GRISOTTO are represented as IUPAC strings, and the scaled Euclidean distance compares only PSSM's, these IUPAC strings need to be converted into PSSM's. However, there are many ways to choose a PSSM to represent an IUPAC symbol and this choice affects the Euclidean distance and, therefore, the final metric. For this reason we should not use an *ad hoc* representation and, instead, should justify the representation with some theoretical foundation.

Herein, we choose the PSSM representation of an IUPAC symbol such that the scaled Euclidean distance between the PSSM representations of two IUPAC symbols, say α and β, is the closest possible to the average scaled Euclidean distance between (any) PSSM's that represent α and β. The alluded distance is presented in Section C.1, while the PSSM representation of an IUPAC string is presented in Section C.2. Upon defining this translation, we consider precisely the same metric used in PRIORITY and MEME, which we describe in Section C.3.

C.1 Minimum scaled Euclidean distance

Consider that both P and Q are PSSM's, that is, P and Q are matrixes of dimension $4 \times k$ such that $\sum_{i=1}^{4} P_{ij} = 1$, for all $1 \leq j \leq k$, where each line represents a letter of the DNA alphabet and each column represents a motif position. The metric used to compute the distance between P and Q was proposed by the PRIORITY researchers (Narlikar et al., 2007) and it is a scaled version of the Euclidean distance. The *scaled Euclidean distance* is such that the maximum distance is 1, and the minimum distance is 0, leading to the following expression:

$$sd(P,Q) = \frac{1}{k} \sum_{j=1}^{k} \sqrt{\sum_{i=1}^{4} \frac{(P_{ij} - Q_{ij})^2}{2}}. \tag{C.1}$$

From (C.1) it is easy to understand that the contribution of the j-th column of P and Q is given by

$$\delta(P_j, Q_j) = \sqrt{\sum_{i=1}^{4} \frac{(P_{ij} - Q_{ij})^2}{2}}. \tag{C.2}$$

Observe that each column of a PSSM consists in a multinomial distribution over the DNA, precisely the same type of information encoded by a IUPAC symbol. In the next subsection we discuss how to convert an IUPAC symbol into a multinomial distribution over the DNA in a meaningful way.

C.2 PSSM representation of an IUPAC string

Each symbol of an IUPAC string will be translated into a column of a PSSM matrix, that is, into a multinomial distribution over the DNA alphabet. Note that each IUPAC symbol has a canonical distribution over the DNA alphabet, which is presented in Table C.1 (page 169).

The canonical distribution of the IUPAC symbols is of little use in practice since it gives zero probability of having mismatches (e.g., observing an A instead of an C in a string), which can lead to irrecoverable errors. It is a common mistake to assign probability zero to a event that is extremely unlikely, but not impossible. Therefore, it is usual to consider a small probability of having mismatches and replace each zero probability in the distributions by a small value ε, denoting an error probability. Finding a meaningful error is not a straightforward task. We devote the rest of this section to this endeavor.

p	p_A	p_C	p_G	p_T
A	1	0	0	0
C	0	1	0	0
G	0	0	1	0
T	0	0	0	1
R	$\frac{1}{2}$	0	$\frac{1}{2}$	0
Y	0	$\frac{1}{2}$	0	$\frac{1}{2}$
M	$\frac{1}{2}$	$\frac{1}{2}$	0	0
K	0	0	$\frac{1}{2}$	$\frac{1}{2}$
W	$\frac{1}{2}$	0	0	$\frac{1}{2}$
S	0	$\frac{1}{2}$	$\frac{1}{2}$	0
B	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
D	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{3}$
H	$\frac{1}{3}$	$\frac{1}{3}$	0	$\frac{1}{3}$
V	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0
N	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

Table C.1: Canonical distribution of the IUPAC symbols

Start by noticing that a multinomial distribution $p = (p_A, p_C, p_G, p_G)$ can be translated into an IUPAC symbol α. Indeed, α should be the symbol whose canonical distribution is closer to p, that is, α should be the symbol that minimizes the distance

$$\delta(p, q^\alpha) = \sqrt{\sum_{i=1}^{4} \frac{(p_i - q_i^\alpha)^2}{2}},$$

where q^α is the canonical distribution of the IUPAC symbol α given in Table C.1. Observe that there are distributions that distance the same from the canonical distributions of two different IUPAC symbols (e.g., $p = (\frac{3}{4}, 0, \frac{1}{4}, 0)$ has the same distance to the canonical distributions of A and R). However, these distributions have probability zero of occurring in practice, that is, they form a measure-zero set. Therefore, they can be disregarded or assumed to be deterministically translated to one of the possible IUPAC symbols. Recall that the set of all multinomial distributions over the DNA alphabet constitutes exactly the standard 4-simplex set Δ^4, that is,

$$\Delta^4 = \left\{ (x_1, x_2, x_3, x_4) \in \mathbb{R}^4 : \sum_{i=1}^{4} x_i = 1 \wedge \bigwedge_{i=1}^{4} x_i \geq 0 \right\}.$$

Thus, IUPAC symbols generate a partition $\{P_\alpha\}_{\alpha \in \Sigma}$ over Δ^4 where

$$P_\alpha = \{p \in \Delta^4 : \delta(p, q^\alpha) < \delta(p, q^\beta) \text{ for all IUPAC symbols } \beta \neq \alpha\}.$$

Now, by assuming that all PSSM's that represent an IUPAC symbol may occur with the same probability and that they occur independently, it is possible to define the distance between two IUPAC symbols α and β as the average distance between a PSSM in P_α with a PSSM in P_β, that is:

$$d(\alpha, \beta) = \int_{p \in P_\alpha} \int_{q \in P_\beta} \frac{1}{|P_\alpha|} \frac{1}{|P_\beta|} \delta(p, q) \, \mathrm{d}p \, \mathrm{d}q \tag{C.3}$$

where $|P_\alpha|$ is the volume of P_α and $|P_\beta|$ is the volume of P_β. We performed a Monte Carlo approximation to the integral (C.3) for all possible values of $p \in P_\alpha$ and $q \in P_\beta$ by generating 10000 PSSM's at random, and obtained in this way numerical approximations of the distances between IUPAC symbols. The Monte Carlo simulation can be found at GRISOTTO webpage. The results are presented in Table C.4 (page 173).

Now, our goal is to translate an IUPAC symbol into a PSSM column in a way that mimics the average distances given in Table C.4. For this purpose, we choose a representative for each set P_α such that the differences between representatives are (as close as possible to) the average difference between the partition sets. The rationale for this approach is that the distance between the representatives of P_α and P_β should be as close as possible to the average distance $d(\alpha, \beta)$ since any PSSM representing α or β may occur.

In order to obtain such translation we consider 3 degrees of freedom on the errors: (i) ε_x, when a DNA symbol is considered; (ii) ε_y, when a degenerate mixture of two symbols is considered; and (iii) ε_z, when a degenerate mixture of three symbols is considered. In detail, we adopt the translation given in Table C.2. Clearly, these degrees of freedom ensure a symmetric IUPAC translation that maps, say, A and C in a similar way, up to a permutation of p_A and p_C. Moreover, the translation of A should be invariant under permutations of p_C, p_G and p_T. A similar desideratum is taken into account for the translation of degenerate

p	p_A	p_C	p_G	p_T
A	$1-3\varepsilon_x$	ε_x	ε_x	ε_x
C	ε_x	$1-3\varepsilon_x$	ε_x	ε_x
G	ε_x	ε_x	$1-3\varepsilon_x$	ε_x
T	ε_x	ε_x	ε_x	$1-3\varepsilon_x$
R	$\frac{1}{2}-\varepsilon_y$	ε_y	$\frac{1}{2}-\varepsilon_y$	ε_y
Y	ε_y	$\frac{1}{2}-\varepsilon_y$	ε_y	$\frac{1}{2}-\varepsilon_y$
M	$\frac{1}{2}-\varepsilon_y$	$\frac{1}{2}-\varepsilon_y$	ε_y	ε_y
K	ε_y	ε_y	$\frac{1}{2}-\varepsilon_y$	$\frac{1}{2}-\varepsilon_y$
W	$\frac{1}{2}-\varepsilon_y$	ε_y	ε_y	$\frac{1}{2}-\varepsilon_y$
S	ε_y	$\frac{1}{2}-\varepsilon_y$	$\frac{1}{2}-\varepsilon_y$	ε_y
B	ε_z	$\frac{1-\varepsilon_z}{3}$	$\frac{1-\varepsilon_z}{3}$	$\frac{1-\varepsilon_z}{3}$
D	$\frac{1-\varepsilon_z}{3}$	ε_z	$\frac{1-\varepsilon_z}{3}$	$\frac{1-\varepsilon_z}{3}$
H	$\frac{1-\varepsilon_z}{3}$	$\frac{1-\varepsilon_z}{3}$	ε_z	$\frac{1-\varepsilon_z}{3}$
V	$\frac{1-\varepsilon_z}{3}$	$\frac{1-\varepsilon_z}{3}$	$\frac{1-\varepsilon_z}{3}$	ε_z
N	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

Table C.2: Distribution of the IUPAC symbols with three types of errors.

symbols. Now we apply the least squares' method to obtain the values of ε_x, ε_y and ε_z that minimize

$$\sum_{\alpha,\beta\in\Sigma} (\delta(q^{\alpha}(\varepsilon_x,\varepsilon_y,\varepsilon_z), q^{\beta}(\varepsilon_x,\varepsilon_y,\varepsilon_z)) - d(\alpha,\beta))^2,$$

where $q^{\alpha}(\varepsilon_x,\varepsilon_y,\varepsilon_z)$ is the distribution of the IUPAC symbol α given in Table C.2 and $d(\alpha,\beta)$ is the distance given by Equation (C.3) computed according to the approximation presented in Table C.4 (page 173). The solution obtained for this problem is

- $\varepsilon_x = 0.0577185$,
- $\varepsilon_y = 0.03827495$ and
- $\varepsilon_z = 0.005683$,

which leads to the final translation given in Table C.3. The detailed calculus of these values by the least squares' method can be found at GRISOTTO webpage.

p	p_A	p_C	p_G	p_T
A	0.826845	0.0577185	0.0577185	0.0577185
C	0.0577185	0.826845	0.0577185	0.0577185
G	0.0577185	0.0577185	0.826845	0.0577185
T	0.0577185	0.0577185	0.0577185	0.826845
R	0.461725	0.03827495	0.461725	0.03827495
Y	0.03827495	0.461725	0.03827495	0.461725
M	0.461725	0.461725	0.03827495	0.03827495
K	0.03827495	0.03827495	0.461725	0.461725
W	0.461725	0.03827495	0.03827495	0.461725
S	0.03827495	0.461725	0.461725	0.03827495
B	0.005683	0.331439	0.331439	0.331439
D	0.331439	0.005683	0.331439	0.331439
H	0.331439	0.331439	0.005683	0.331439
V	0.331439	0.331439	0.331439	0.005683
N	0.25	0.25	0.25	0.25

Table C.3: Translation of IUPAC symbols whose distance is closer to the average distance.

In the rest of this work, we assume that the PSSM column that represents an IUPAC symbol is given by Table C.3. Moreover, such translation is used to convert the IUPAC string representing the literature motif and the motif reported by GRISOTTO.

$d(\cdot,\cdot)$	A	C	G	T	R	Y	M	K	W	S	B	D	H	V	N
A	0.117773	0.78676	0.78676	0.78676	0.401808	0.693145	0.693145	0.401808	0.693145	0.401808	0.652724	0.467727	0.467727	0.467727	0.494578
C	0.78676	0.117773	0.78676	0.78676	0.693145	0.401808	0.693145	0.401808	0.401808	0.693145	0.467727	0.652724	0.467727	0.467727	0.494578
G	0.78676	0.78676	0.117773	0.78676	0.401808	0.693145	0.401808	0.693145	0.401808	0.693145	0.467727	0.467727	0.652724	0.467727	0.494578
T	0.78676	0.78676	0.78676	0.117773	0.467727	0.401808	0.401808	0.467727	0.467727	0.401808	0.467727	0.467727	0.467727	0.652724	0.494578
R	0.401808	0.693145	0.401808	0.467727	0.195582	0.585482	0.429646	0.429646	0.429646	0.429646	0.43141	0.294668	0.43141	0.294668	0.332181
Y	0.693145	0.401808	0.693145	0.401808	0.585482	0.195582	0.429646	0.429646	0.429646	0.429646	0.294668	0.43141	0.294668	0.43141	0.332181
M	0.693145	0.693145	0.401808	0.401808	0.429646	0.429646	0.195582	0.585482	0.429646	0.429646	0.294668	0.294668	0.43141	0.43141	0.332181
K	0.401808	0.401808	0.693145	0.467727	0.429646	0.429646	0.585482	0.195582	0.429646	0.429646	0.43141	0.43141	0.294668	0.294668	0.332181
W	0.693145	0.401808	0.401808	0.467727	0.429646	0.429646	0.429646	0.429646	0.195582	0.585482	0.294668	0.43141	0.43141	0.294668	0.332181
S	0.401808	0.693145	0.693145	0.401808	0.429646	0.429646	0.429646	0.429646	0.585482	0.195582	0.43141	0.294668	0.294668	0.43141	0.332181
B	0.652724	0.467727	0.467727	0.467727	0.43141	0.294668	0.294668	0.43141	0.294668	0.43141	0.196942	0.319852	0.319852	0.319852	0.244875
D	0.467727	0.652724	0.467727	0.467727	0.294668	0.43141	0.294668	0.43141	0.43141	0.294668	0.319852	0.196942	0.319852	0.319852	0.244875
H	0.467727	0.467727	0.652724	0.467727	0.43141	0.294668	0.43141	0.294668	0.43141	0.294668	0.319852	0.319852	0.196942	0.319852	0.244875
V	0.467727	0.467727	0.467727	0.652724	0.294668	0.43141	0.43141	0.294668	0.294668	0.43141	0.319852	0.319852	0.319852	0.196942	0.244875
N	0.494578	0.494578	0.494578	0.494578	0.332181	0.332181	0.332181	0.332181	0.332181	0.332181	0.244875	0.244875	0.244875	0.244875	0.181994

Table C.4: Average distance between PSSM's representing IUPAC symbols.

C.3 Best alignment and cutoffs

Now that we have justified the translation of IUPAC strings into PSSM's, we can assess the accuracy of GRISOTTO by using exactly the same criterion as in PRIORITY (Narlikar et al., 2007; Gordân et al., 2008; Gordân and Hartemink, 2008; Gordân et al., 2010) and MEME (Bailey et al., 2010). In this section we discuss this criterion in detail.

As discussed in Section C.1, PRIORITY researchers considered the scaled Euclidean distance (as in Equation (C.1)) to measure the mismatch between the PSSM of the documented motif, say P, and the reported one, say Q. However, the reported motif may be of different size or in the opposite DNA strand relatively to the documented one. To address this issue the distance between P and Q was considered to be the *minimum scaled Euclidean distance* for all possible alignments, over an overlap window, of the reported motif (or its reverse complement) with the known motif. Overlap windows of size ranging from $\min(6, \dim(P), \dim(Q))$ to $\min(\dim(P), \dim(Q))$ are considered.

Following Narlikar et al. (2007); Gordân et al. (2008); Gordân and Hartemink (2008); Gordân et al. (2010); Bailey et al. (2010), the top scoring motif correctly predicts the literature one if this minimum scaled Euclidean distance for all alignments (considering also the reverse complement) is smaller than 0.25, being enough to have a matching overlap window of size 6. However, this distance is considered only if the average information content per position of the reported motif is at least 1 bit and the distance between columns is at most 0.8, otherwise the distance between motifs is 1. PRIORITY researchers called into attention that such cutoffs (minimum distance 0.25, average entropy 1 and minimum column distance 0.8) are probably imperfect but were chosen to automate the evaluation process and to reduce the possibility of introducing a subjective bias into the results. Moreover, the authors argued that relative results of all evaluated algorithms are generally insensitive to a range of reasonable choices of these cutoffs. In order to make our results directly comparable with the results from PRIORITY and MEME we used exactly the same metric.

The criterion discussed above was implemented and made available in a Perl script by PRIORITY researchers (Gordân et al., 2008). We translated this Perl script to pseudocode in Algorithm C.1 and incorporated it in the Java source of GRISOTTO. At the light of the previous discussion the algorithm is self-explanatory.

Algorithm C.1 ComputeDistance, minimum scaled Euclidean distance with cutoffs

```
ComputeDistance(PSSM P, PSSM Q)
 1. reversed := false;
 2. if (length($P$) > length($Q$)) then
 3.   $(P, Q) := (Q, P)$;
 4.   reversed := true; // make P the matrix with less columns and mark if the reverse was made
 5. $R$ := DNA_complement ($P$); // let $R$ be the PSSM denoting the reverse DNA complement of $P$
 6. overlap := $\min$(6,length($P$)); // this is the minimum overlap considered to compute the distance
 7. dist := $+\infty$;
 8. for len := overlap to length($P$) do // for all possible overlaps
 9.   for j1 := 1 to length($P$)$-$len+1 do // for all possible starting positions of $P$
10.     for j2 := 1 to length($Q$)$-$len+1 do // for all possible starting positions of $Q$
11.       (sumPQ,sumRQ) := (0,0); // initialize the variable that store the sums to 0
12.       (cdPQ,cdRQ) := (0,0); // initialize the variable that control the columns distance to 0
13.       Ent := 0; // initialize the variable that control the entropy of the reported motif to 0
14.       for j := 0 to len$-1$ do // for each column of the current alignment
15.         (sPQ,sRQ) := (0,0); // initialize the contribution of each column to the distance to 0
16.         for i := 1 to 4 do // for each row ranging in the DNA alphabet
17.           sPQ + := $(P_{i(j_1+j)} - Q_{i(j_2+j)})^2$; // contribution of the current row and column
18.           sRQ + := $(R_{i(j_1+j)} - Q_{i(j_2+j)})^2$; // contribution in DNA complement of the current row and column
19.           if (reversed) then val := $Q_{i(j_2+j)}$; // if Q is the reported motif
20.           else val := $P_{i(j_i+j)}$; // if P is the reported motif
21.           if (val = 0) then val := $0.001$; // avoid log 0 problems
22.           Ent + := val$\times \log_2$(val); // compute the entropy of the column
23.         if ($\sqrt{(\mathrm{sPQ}/2)}$ >0.8) then cdPQ++; // distance of any column should not be greater than 0.8
24.         if ($\sqrt{(\mathrm{sRQ}/2)}$ >0.8) then cdRQ++; // distance of any column should not be greater than 0.8
25.         sumPQ + := $\sqrt{\mathrm{sPQ}}$; // distance contribution of the column
26.         sumRQ + := $\sqrt{\mathrm{sRQ}}$; // distance contribution of the column
27.       Ent := 2+Ent/len; // entropy update
28.       if (cpPQ/len $\geq \frac{1}{6}$ or Ent $\leq 1$) then dPQ := 1; // not enough information in P
29.       else dPQ := sumPQ/(len$\times\sqrt{2}$);
30.       if (dPQ < dist) then dist := dPQ;
31.       if (cpRQ/len $\geq \frac{1}{6}$ or Ent $\leq 1$) then dRQ := 1; // not enough information in R
32.       else dRQ := sumPQ/(len$\times\sqrt{2}$);
33.       if (dRQ < dist) then dist := dRQ;
34. return dist;
```

Appendix D

Evaluating various positional priors

This chapter makes the results presented in Section 7.3 (page 138) reproducible along with the data and algorithms provided in the GRISOTTO webpage.[1] We start by presenting pertinent information about the evaluation methodology, including, parameter settings (Section D.1) and running times (Section D.2). Finally, in Section D.3 we discuss in detail results obtained in the experimental methodology and finish with follow-up work.

D.1 Parameter settings

In Table 7.2 (page 141), we compare the results of GRISOTTO with the results of twelve state-of-the-art motif discoverers: PhyloCon (Wang and Stormo, 2003), PhyME (Sinha et al., 2004), MEME (Bailey and Elkan, 1995b), MEME_c (Harbison et al., 2009), PhyloGibbs (Siddharthan et al., 2005), Kellis *et al.* (Kellis et al., 2003), CompareProspector (Liu et al., 2004), Converge (MacIsaac et al., 2006), MEME-$\mathcal{DC}$ (Bailey et al., 2010), PRIORITY-$\mathcal{DC}$ (Gordân et al., 2008, 2010), PRIORITY-$\mathcal{DE}$ (Gordân and Hartemink, 2008), PRIORITY-$\mathcal{DN}$ (Narlikar et al., 2007). Of the twelve methods considered in our analysis we used the results reported by Gordân et al. (2010) and Bailey et al. (2010). Parameter settings for these methods can be found in the supplementary material of the original papers.

Next, we provide all parameters (empirically computed) that were used to run the 156 yeast ChiP-chip experiments, making in this way the results reproducible. RISOTTO was tuned, as described in Section B, with default parameters, that is, q-step $= 5$, q-min $= 5$ and

[1] `http://kdbio.inesc-id.pt/~asmc/software/grisotto.html`

q-max $= 100$. GRISOTTO used $z = z_{min} = 50$ and $z_{max} = 80$, that is, GRISOTTO asked for an output of RISOTTO between 50 and 80 motifs and post-processed only 50. As mentioned in the Section 7.3 (page 138) $k = 8$ as priors were also devised for 8-mers. The output of RISOTTO depends on one more parameter, one that is passed to SMILE shuffling-based statistical significance procedure. This procedure needs the size of the shuffling-mer, we used always 6. We notice that this shuffling procedure depends on a seed, that is, different seeds may give rise to different outputs. Of course, if the same seed is not used in the experiments negligible differences in the results may occur. In GRISOTTO webpage we provide the seed used. Moreover, if one does not want to use RISOTTO, and SMILE statistical significance, we also provide the exact output of RISOTTO used in the 156 experiments. In this way, to reproduce the results, the user only needs to download and run GRISOTTO. Actually, GRISOTTO only computes RISOTTO output one time for each sequence-set, reusing it in the following runs until RISOTTO parameters change. If the output for the new parameters are not available (each RISOTTO output is stored in a different folder whose name identifies the parameters used) then RISOTTO is called, otherwise it is used the output from the respective folder.

It remains to detail the parameters used to balance the priors. GRISOTTO-$\mathcal{DC}$ used $\lambda = \frac{2}{23}$ (as only one prior is considered, $\alpha_{\mathcal{DC}} = 1$). This corresponds to giving 10.5 more weight to the $\mathcal{DC}$ prior than to the over-representation of the motifs in the DNA sequences. GRISOTTO-$\mathcal{DE}$ used $\lambda = \frac{2}{15}$, indicating that $\mathcal{DE}$ prior weights 6.5 times more than over-representation. GRISOTTO-$\mathcal{DN}$ used $\lambda = \frac{1}{7}$, indicating that $\mathcal{DN}$ prior weights 6 times more than over-representation. Finally, GRISOTTO-$\mathcal{CDP}$ used $\lambda = \frac{1}{21}$ and $\alpha_{\mathcal{DC}} = \frac{2}{5}$, $\alpha_{\mathcal{DE}} = \frac{7}{20}$ and $\alpha_{\mathcal{DN}} = \frac{1}{4}$. This testify that $\mathcal{DC}$ prior weights 8, $\mathcal{DE}$ prior weights 7 and $\mathcal{DN}$ prior weights 5 times more than over-representation of the motifs in the DNA promoter sequences, respectively.

Concerning the 13 mouse ChiP-seq data, the RISOTTO was tuned as for the yeast data. Moreover, GRISOTTO-$\mathcal{DC}$ used exactly the same λ as for the yeast data (as only one prior is considered $\alpha_{\mathcal{DC}} = 1$). For the coverage-based prior we used $\lambda = \frac{1}{2}$ as we believe it contains chiefly as many information as overrepresentation. When combining the coverage-based prior with the $\mathcal{DC}$ prior, we tried several weights for λ and α's, including the uniform weight between priors, however the results were exactly the same as if we only consider the single $\mathcal{DC}$ prior.

D.2 Running times

When running the yeast ChiP-chip experiments we noticed that GRISOTTO rarely reported IUPAC strings with degenerate symbols of the IUPAC code (that is, IUPAC except DNA). This made us try to search for motifs using just the DNA alphabet. Indeed, results presented in Table 7.2 (page 141) considered only the DNA alphabet and coincide with those using the full IUPAC alphabet. This boosted significantly the time of the algorithm. Using DNA alphabet, GRISOTTO was able to report all 156 top scoring motifs within 5-6 minutes using a standard machine (one core of a Intel 2.4 GHz core 2 Duo), taking around 2-3 seconds per sequence-set. This time includes running RISOTTO algorithm and computing the distance of the reported motif to the documented one.

The average number of nucleotides on yeast ChiP-chip sequence-sets is around 100.000, whereas for the mouse ChiP-seq data is around 4.000.000, therefore, for the mouse data GRISOTTO took 1-4 minutes per sequence-set. In closing, we emphasize that GRISOTTO was able to use all sequences in each of the 13 mouse sequence-sets. The same experiments performed by MEME used only 100 randomly chosen sequences for each sequence-set, working in this way with only around 200.000bp per sequence-set.

D.3 Detailed results

Herein, we further detail the results presented in Table 7.2 (page 141). The intended reader should refer to Section 7.3 to find experimental methodology, including, sequence-sets and PSP's used in the experiments. A table comparing the results of GRISOTTO and PRIORITY using various positional priors can be found at the GRISOTTO webpage. Therein, it can be found details about which motif was correctly predicted, sequence-set by sequence-set, by both algorithms. In the following we use this table to provide a closer inspection over the results presented in Section 7.3.

The analysis of the aforementioned table was decisive to encourage us to combine priors from different sources as we found that individual priors, although having some degree of redundancy, still report many disjoint motifs. As an example, although $\mathcal{DE}$ and $\mathcal{DN}$ correctly predicted almost the same number of motifs out of the 156 experiments, in 29 of these experiments, only one of the two succeeded (including sequence-sets 9, 17, 19, 20, 21, 23, 26, 34, 35, 42, 53, 59, 61, 75, 77, 81, 89, 96, 101, 117, 119, 120, 121, 125, 129, 136, 137, 143, 156). Indeed, GRISOTTO-$\mathcal{DE}$ found 16 motifs that GRISOTTO-$\mathcal{DN}$ did not, whereas GRISOTTO-$\mathcal{DN}$ found 13 motifs that GRISOTTO-$\mathcal{DE}$ did not. Moreover, if we conduct a closer inspection over these 29 sequence-sets, we conclude that GRISOTTO-$\mathcal{DC}$ fails in 11 out of these 29 (including sequence-sets 9, 20, 23, 35, 53, 75, 77, 81, 89, 101, 143). This suggests that combining the priors has potential to improve motif discoverers, more likely, on those 11 sequence-sets (as other 18 already have two priors up-weighting the true motif). There could be, however, other improvements as a motif might not be found by the priors, when individually considered, but it might be unraveled from the convex closure of the information given by them.

By analyzing the results of GRISOTTO-$\mathcal{CDP}$ we check that from those 29 sequence-sets, where only one of GRISOTTO-$\mathcal{DE}$ and GRISOTTO-$\mathcal{DN}$ succeeded, GRISOTTO-$\mathcal{CDP}$ failed in 11 (including sequence-sets 9, 20, 21, 23, 35, 42, 75, 81, 89, 101, 125). Moreover, from these 11 sequence-sets, GRISOTTO-$\mathcal{DC}$ also failed to unravel 8 motifs (including sequence-sets 9, 20, 23, 35, 75, 81, 89, 101). This means that GRISOTTO-$\mathcal{CDP}$ was not able to find only 3 motifs that were being up-weighted by two priors (including sequence-sets 21, 42, 125). Finally, we mentioned above that there could be cases where priors, when individually considered, may fail in giving extra information for motif discovery, but may succeed when combined together. In practice, this was the case of 7 sequences-sets (including sequence-sets 2, 32, 87, 91, 116, 133, 146, 154) where only GRISOTTO-$\mathcal{CDP}$ succeeded. This was for sure a great advantage of GRISOTTO-$\mathcal{CDP}$ relatively to GRISOTTO-$\mathcal{DC}$, GRISOTTO-$\mathcal{DE}$ and GRISOTTO-$\mathcal{DN}$.

Next, we analyze the relative results of GRISOTTO when $\mathcal{DC}$, $\mathcal{DE}$ and $\mathcal{DN}$ priors are considered individually and when combined. Firstly, GRISOTTO-$\mathcal{DC}$ was able to discover 9 motifs that $\mathcal{DE}$ and $\mathcal{DN}$ were unable to characterize. Similarly, GRISOTTO-$\mathcal{DE}$ was able to find 6 and GRISOTTO-$\mathcal{DN}$ 5, that the other two were not. Hence, the number of motifs that are only characterized by one of the priors amount to 20. Moreover, 104 motifs were correctly predicted by GRISOTTO with at least one of $\mathcal{DC}$, $\mathcal{DE}$ and $\mathcal{DN}$ priors. Knowing that GRISOTTO-$\mathcal{CDP}$ was able to correctly predict 93 motifs (refer to Table 1 in main text), and that 7 of these motifs were not unraveled by any of the priors when individually considered (refer to the previous paragraph), we deduce that GRISOTTO-$\mathcal{CDP}$ mislaid 18 motifs that were previously found by at least one of GRISOTTO-$\mathcal{DC}$, GRISOTTO-$\mathcal{DE}$ and GRISOTTO-$\mathcal{CDP}$. This shows that although the combination of priors does not recover all the motifs found by at least one of the combined priors, it also unravels some novel motifs that none of the priors were able to find separately.

Moreover, we also evaluate the overall results obtained by GRISOTTO and PRIORITY when all priors are considered. Having this in mind, it is worthwhile noticing that both motif discoverers succeeded with all $\mathcal{DC}$, $\mathcal{DE}$, $\mathcal{DN}$ and $\mathcal{CDP}$ priors in 50 sequence-sets. That is, $\mathcal{DC}$, $\mathcal{DE}$, $\mathcal{DN}$ and $\mathcal{CDP}$ priors were able to find 50 motifs from the 156 sequence-sets, independently from the discoverer that was used ($\mathcal{CDP}$ was only tested within GRISOTTO). If we count only motifs correctly predicted by both discoverers at least with one of these priors, but not by all priors, the number of correct predictions is 31. Moreover, PRIORITY failed, with all priors considered, whereas GRISOTTO succeeded with at least one of the priors in 31 sequences-sets. On the other hand, GRISOTTO failed, with all priors considered, while PRIORITY succeeded with at least one of the priors in only 1 sequence-set (we acknowledge that GRISOTTO was evaluated with $\mathcal{CDP}$ while we do not have the means to do the same with PRIORITY). Finally, both GRISOTTO and PRIORITY failed, with all priors considered, in 43 sequences-sets. Therefore, GRISOTTO and PRIORITY together, by considering all available priors, were able to unravel 113 motifs out of the 156 experiments (as they all fail in 43). Moreover, GRISOTTO was able to recover 112 out of these 113 whereas PRIORITY only discovered 82.

Although it is natural that different algorithms with different PSP's unravel a disjoint set of motifs from the 156 sequence-sets, it is interesting to notice that both algorithms failed to discover 43 motifs. In the following we disclose which motifs are these. We classify these 43 sequence-sets in four different categories: (i) motifs with spacers, that is, motifs with at least three consecutive N IUPAC symbols in the middle of the consensus string (failed in 7 out of 11 sequence-sets, namely, sequence-sets 31, 33, 90, 122, 128, 139, 140); (ii) motifs longer than, or equal to, 8 sites, excluding the ones with spacers (failed in 9 out of 70 sequence-sets, namely, sequence-sets 7, 22, 41, 67, 70, 102, 103, 106, 109); (iii) motifs shorter than 8 sites with no mismatches (failed in 17 out of 51 sequence-sets, namely, sequence-sets 8, 43, 44, 85, 86, 100, 110, 111, 112, 113, 114, 115, 131, 148, 149, 150, 151) and (iv) motifs shorter than 8 sites with mismatches (failed in 10 out of 24 sequence-sets, mainly, sequence-sets 3, 4, 71, 72, 73, 76, 78, 79, 80, 135). We add that motifs with spacers listed in (i) have minimum size of 10. We conclude that $\mathcal{DC}$, $\mathcal{DE}$, $\mathcal{DN}$ and $\mathcal{CDP}$ were not able to characterize 64% of the motifs with spacers, and 42% of the motifs with degenerate symbols shorter than 8 sites. This strongly suggests that priors considering 8-mers fail to characterize motifs shorter than 8 sites, specially when they are not highly conserved. Moreover, motifs with spacers are also not suitably described by the considered priors.

Follow-up work should include: (i) studying the effect of prior-mers size in motif discovery; (ii) understanding how to build better priors for motifs with spacers (priors were not able to describe much more than half of the cases present in the sequence-sets) (iii) devising new priors from different sources and combining them with existing ones into the BIS score.

Bibliography

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

J. Allali. Structures d'indexation: Les arbres des facteurs. Memoire de maitrise, University of Marne-la-Vallée, 2000.

J. Allali and M.-F. Sagot. The at most k-deep factor tree. Technical Report 2004-03, Institut Gaspard Monge, Université Marne la Vallée, 2004.

T. L. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proc. ISMB'94*, pages 28–36, 1994.

T. L. Bailey and C. Elkan. Unsupervised learning of multiple motifs in biopolymers using EM. *Machine Learning*, 21(1-2):51–80, 1995a.

T. L. Bailey and C. Elkan. The value of prior knowledge in discovering motifs with MEME. In *Proc. ISMB'95*, pages 21–29, 1995b.

T. L. Bailey, M. Bodén, T. Whitington, and P. Machanick. The value of position-specific priors in motif discovery using MEME. *BMC Bioinformatics*, 11:179, 2010.

Y. Barash, G. Elidan, N. Friedman, and T. Kaplan. Modeling dependencies in protein-DNA binding sites. In *Proc. RECOMB'03*, pages 28–37, 2003.

A. R. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, 1998.

R. G. Beiko and R. L. Charlebois. GANN: Genetic algorithm neural networks for the detection of conserved combinations of features in DNA. *BMC Bioinformatics*, 6:36, 2005.

A. J. Bell. The co-information lattice. In *Proc. ICA'03*, pages 921–926, 2003.

P. V. Benos, M. L. Bulyk, and G. D. Stormo. Additivity in protein-DNA interactions: How good an approximation is it? *Nucleic Acids Research*, 30(20):4442–4451, 2002.

J. Bilmes. Dynamic Bayesian multinets. In *Proc. UAI'00*, pages 38–45, 2000.

R. R. Bouckaert. *Bayesian Belief Networks: From Construction to Inference.* PhD thesis, University of Utrecht, 1995.

A. Brazma, I. Jonassen, I. Eidhammer, and D. Gilbert. Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology*, 5(2):279–305, 1998a.

A. Brazma, I. Jonassen, J. Vilo, and E. Ukkonen. Predicting gene regulatory elements *in silico* on a genomic scale. *Genome Research*, 8(11):1202–1215, 1998b.

J. Buhler and M. Tompa. Finding motifs using random projections. *Journal of Computational Biology*, 9(2):225–242, 2002.

W. L. Buntine. Theory refinement on Bayesian networks. In *Proc. UAI'91*, pages 52–60, 1991.

L. R. Cardon and G. D. Stormo. Expectation maximization algorithm for identifying protein-binding sites with variable length from unaligned DNA fragments. *Journal of Molecular Biology*, 223(1):159–170, 1992.

A. M. Carvalho. Efficient algorithms for structured motif extraction in DNA sequences. Master's thesis, IST, UTL, 2004. Supervised by A. L. Oliveira and A. T. Freitas.

A. M. Carvalho. Scoring functions for learning Bayesian networks. Technical report, INESC-ID Tec. Rep. 54/2009, 2009.

A. M. Carvalho and A. L. Oliveira. Learning Bayesian networks consistent with the optimal branching. In *Proc. ICMLA'07*, pages 369–374, 2007.

A. M. Carvalho and A. L. Oliveira. GRISOTTO: A greedy approach to improve combinatorial algorithms for motif discovery with prior knowledge. *Algorithms for Molecular Biology*, 6: 13, 2011.

A. M. Carvalho, A. T. Freitas, A. L. Oliveira, and M.-F. Sagot. Efficient extraction of structured motifs using box-links. In Alberto Apostolico and Massimo Melucci, editors, *Proc. SPIRE'04*, volume 3246 of *LNCS*, pages 267–268. 2004.

A. M. Carvalho, A. T. Freitas, A. L. Oliveira, and M.-F. Sagot. A highly scalable algorithm for the extraction of cis-regulatory regions. In Yi-Ping Phoebe Chen and Limsoon Wong, editors, *Proc. APBC'05*, volume 1 of *ABCB*, pages 273–282. 2005.

A. M. Carvalho, A. T. Freitas, A. L. Oliveira, and M.-F. Sagot. An efficient algorithm for the identification of structured motifs in DNA promoter sequences. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(2):126–140, 2006.

A. M. Carvalho, A. L. Oliveira, and M.-F. Sagot. Efficient learning of Bayesian network classifiers: An extension to the TAN classifier. In M. A. Orgun and J. Thornton, editors, *Proc. IA'07*, volume 4830 of *LNCS*, pages 16–25. 2007.

A. M. Carvalho, T. Roos, A. L. Oliveira, and P. Myllymäki. Discriminative learning of Bayesian networks via factorized conditional log-likelihood. *Journal of Machine Learning Research*, 12, 2011. In print.

S. Cawley. *Statistical models for DNA sequencing and analysis.* PhD thesis, Berkely, University of California, 2000.

X. Chen, H. Xu, F. Yuan, P. Fang, M. Huss, V. B. Vega, E. Wong, Y. L. Orlov, W. Zhang, J. Jiang, Y.-H. Loh, H. C. Yeo, Z. X. Yeo, V. Narang, K. R. Govindarajan, B. Leong, A. Shahab, Y. Ruan, G. Bourque, W.-K. Sung, N. D. Clarke, C.-L. Wei, and H.-H. Ng. Integration of external signaling pathways with the core transcriptional network in embryonic stem cells. *Cell*, 133(6):1106–1117, 2008.

D. M. Chickering. A transformational characterization of equivalent Bayesian network structures. In *Proc. UAI'95*, pages 87–98, 1995.

D. M. Chickering. *Learning Bayesian networks is NP-Complete*, pages 121–130. Learning from data: AI and statistics V. Springer, 1996.

D. M. Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

T. Cover and J. Thomas. *Elements of information theory*. John Wiley & Sons, 2006.

M. Crochemore and M.-F. Sagot. Motifs in sequences: Localization and extraction. In *Compact Handbook of Computational Biology*, pages 47–97. CRC Press, 2004.

F. Daenen, F. van Roy, and P. J. De Bleser. Low nucleosome occupancy is encoded around functional human transcription factor binding sites. *BMC Genomics*, 9(332), 2008.

S. Dasgupta. Learning polytrees. In *Proc. UAI'99*, pages 134–141, 1999.

L. M. de Campos. A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, 7: 2149–2187, 2006.

C. Deremble and R. Lavery. Macromolecular recognition. *Current Opinion in Structural Biology*, 15:171–175, 2005.

P. Domingos and M. J. Pazzani. Simple Bayesian classifiers do not assume independence. In *Proc. AAAI/IAAI'96, Vol. 2*, page 1386, 1996a.

P. Domingos and M. J. Pazzani. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In *Proc. ICML'96*, pages 105–112, 1996b.

P. Domingos and M. J. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2–3):103–130, 1997.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

L. Duret and P. Bucher. Searching for regulatory elements in human noncoding sequences. *Current Opinions in Structural Biology*, 7(3):399–406, 1997.

J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240, 1967.

K. Ellrott, C. Yang, F. M. Sladek, and T. Jiang. Identifying transcription factor binding sites through Markov chain optimization. In *Proc. ECCB'02*, pages 100–109, 2002.

E. Eskin and P. A. Pevzner. Finding composite regulatory patterns in DNA sequences. *Bioinformatics*, 18(1):354–363, 2002.

E. Eskin, U. Keich, M. S. Gelfand, and P. A. Pevzner. Genome-wide analysis of bacterial promoter regions. In *Proc. PSB'03*, pages 29–40, 2003.

U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. IJCAI'93*, pages 1022–1029, 1993.

A. P. Fejes, G. Robertson, M. Bilenky, R. Varhol, M. Bainbridge, and S. J. M. Jones. FindPeaks 3.1: A tool for identifying areas of enrichment from massively parallel short-read sequencing technology. *Bioinformatics*, 24(15):1729–1730, 2008.

Y. M. Fraenkel, Y. Mandel, D. Friedberg, and H. Margalit. Identification of common motifs in unaligned DNA sequences: Application to E*scherichia coli* Lrp regulon. *Computer Applications in Biosciences*, 11(4):379–387, 1995.

N. Friedman and D. Koller. Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1-2):95–125, 2003.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

W. Fu, P. Ray, and E. P. Xing. DISCOVER: A feature-based discriminative method for motif search in complex genomes. *Bioinformatics*, 25(12), 2009.

D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, 82(1-2):45–74, 1996.

R. Gordân and A. J. Hartemink. Using DNA duplex stability information for transcription factor binding site discovery. In *Proc. PSB'08*, pages 453–464, 2008.

R. Gordân, L. Narlikar, and A. J. Hartemink. A fast, alignment-free, conservation-based method for transcription factor binding site discovery. In *Proc. RECOMB'08*, pages 98–111, 2008.

R. Gordân, L. Narlikar, and A. J. Hartemink. Finding regulatory DNA motifs using alignment-free evolutionary conservation information. *Nucleic Acids Research*, 38(6):e90, 2010.

R. Greiner and W. Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. In *Proc. AAAI/IAAI'02*, pages 167–173, 2002.

R. Greiner, X. Su, B. Shen, and W. Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Machine Learning*, 59(3):297–322, 2005.

D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *Proc. ICML'04*, pages 46–53, 2004.

P. D. Grünwald. *The Minimum Description Length Principle.* MIT Press, 2007.

D. Gusfield. *Algorithms on strings, trees, and sequences.* Cambridge University Press, 1997.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

C. T. Harbison, D. B. Gordon, T. I. Lee, N. J. Rinaldi, K. D. Macisaac, T. W. Danford, N. M. Hannett, J. B. Tagne, D. B. Reynolds, J. Yoo, E. G. Jennings, J. Zeitlinger, D. K. Pokholok, M. Kellis, P. A. Rolfe, K. T. Takusagawa, E. S. Lander, D. K. Gifford, E. Fraenkel, and R. A. Young. Transcriptional regulatory code of a eukaryotic genome. *Nature*, 431(7004): 99–104, 2009.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning.* Springer, 2003.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003.

M. Hu, J. Yu, J. M. Taylor, A. M. Chinnaiyan, and Z. S. Qin. On the detection and refinement of transcription factor binding sites using ChiP-seq data. *Nucleic Acids Research*, 38(7): 2154–2167, 2010.

L. C. K. Hui. Color set size problem with applications to string matching. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Proc. CPM'92*, volume 644 of *LNCS*, pages 230–243. 1992.

A. Jakulin. *Machine Learning Based on Attribute Interactions.* PhD thesis, 2005.

S. Karlin, F. Ost, and B. E. Blaisdell. Patterns in DNA and amino acid sequences and their statistical significance. In M. S. Waterman, editor, *Mathematical Methods for DNA Sequences*, pages 133–158. CRC Press, 1989.

K. J. Kechris, E. van Zwet, P. J. Bickel, and M. B. Eisen. Detecting DNA regulatory motifs by incorporating positional trends in information content. *Genome Biology*, 5(7):R50, 2004.

M. Kellis, N. Patterson, M. Endrizzi, B. Birren, and E. S. Lander. Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature*, 423:241–254, 2003.

C. Kirchhamer, C. Yuh, and E. Davidson. Modular cis-regulatory organization of developmentally expressed genes: Two genes transcribed territorially in the sea urchin embryo, and additional examples. In *Proc. Natl. Acad. Sci. USA*, volume 93, pages 9322–9328, 1996.

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. IJCAI'95*, pages 1137–1145, 1995.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97 (1-2):273–324, 1997.

D. Koller and N. Friedman. *Probabilistic Graphical models: Principles and techniques.* MIT Press, 2009.

P. Kontkanen and P. Myllymäki. A linear-time algorithm for computing the multinomial stochastic complexity. *Information Processing Letters*, 103(6):227–233, 2007.

P. Kontkanen, W. Buntine, P. Myllymäki, J. Rissanen, and H. Tirri. Efficient computation of stochastic complexity. In *Proc. AISTATS'03*, pages 233–238, 2003.

I. V. Kulakovskiy, V. A. Boeva, A. V. Favorov, and V. J. Makeev. Deep and wide digging for binding motifs in ChiP-seq data. *Bioinformatics*, 26(20):2622–2623, 2010.

S. Kurtz. Reducing the space requirement of suffix trees. *Software: Practice and Experience*, 29(13):1149–1171, 1999.

I. Lafontaine and R. Lavery. Optimization of nucleic acid sequences. *Biophysical Journal*, 79 (2):680–685, 2000.

I. Lafontaine and R. Lavery. ADAPT: A molecular mechanics approach for studying the structural properties of long DNA sequences. *Biopolymers (Nucleic Acid Science)*, 56: 292–310, 2001a.

I. Lafontaine and R. Lavery. High-speed molecular mechanics searches for optimal DNA interaction sites. *Combinatorial Chemistry & High Throughput Screen*, 4(8):707–717, 2001b.

W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–294, 1994.

A. D. Lanterman. Schwarz, Wallace, and Rissanen: Intertwining themes in theories of model selection. *IEEE Transactions on Information Theory*, 69(2):185–212, 2001.

E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Dover, 1976.

C. Lee, Y. Shibata, B. Rao, B. Strahl, and J. Lieb. Evidence for nucleosome depletion at active regulatory regions genome-wide. *Nature Genetics*, 36(8), 2004.

H.-L. Li and C.-J. Fu. A linear programming approach for identifying a consensus sequence on DNA sequences. *Bioinformatics*, 21(9):1838–1845, 2005.

L. P. Lim and C. B. Burge. A computational analysis of sequence features involved in recognition of short introns. *Proc. Natl. Acad. Sci. USA*, 98(20):11193–11198, September 2001.

Y. Liu, S. Liu, L. Wei, R. B. Altman, and S. Batzoglou. Eukaryotic regulatory element conservation analysis and identification using comparative genomics. *Genome Research*, 14:451–458, 2004.

K. D. MacIsaac, T. Wang, D. B. Gordon, D. K. Gifford, G. D. Stormo, and E. Fraenkel. An improved map of conserved regulatory sites for S*accharomyces cerevisiae*. *BMC Bioinformatics*, 7:113, 2006.

L. Marsan and M.-F. Sagot. Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. *Journal of Computational Biology*, 7(3–4):345–362, 2000.

V. Matys, O. V. Kel-Margoulis, E. Fricke, I. Liebich, S. Land, A. Barre-Dirrie, I. Reuter, D. Chekmenev, M. Krull, K. Hornischer, N. Voss, P. Stegmaier, B. Lewicki-Potapov, H. Saxel, A. E. Kel, and E. Wingender. TRANSFACompel: Transcriptional gene regulation in eukaryotes. *Nucleic Acids Research*, 34(Database-Issue):108–110, 2006.

E. McCreight. A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23(2):262–272, 1976.

W. J. McGill. Multivariate information transmission. *Psychometrika*, 19:97–116, 1954.

C. Meek. Finding a path is harder than finding a tree. *Journal of Artificial Intelligence Research*, 15:383–389, 2001.

M. Meila and M. I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.

P. T. Monteiro, N. D. Mendes, M. C. Teixeira, S. d'Orey, S. Tenreiro, N. P. Mira, H. Pais, A. P. Francisco, A. M. Carvalho, A. B. Lourenço, I. Sá-Correia, A. L. Oliveira, and A. T. Freitas. YEASTRACT-DISCOVERER: New tools to improve the analysis of transcriptional regulatory associations in S*accharomyces cerevisiae*. *Nucleic Acids Research*, 36(Database-Issue):132–136, 2008.

L. Narlikar, R. Gordân, U. Ohler, and A. J. Hartemink. Informative priors based on transcription factor structural class improve *de novo* motif discovery. In *Proc. ISMB'06 (Supplement of Bioinformatics)*, pages 384–392, 2006.

L. Narlikar, R. Gordân, and A. J. Hartemink. Nucleosome occupancy information improves *de novo* motif discovery. In *Proc. RECOMB'07*, pages 107–121, 2007.

D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

R. A. O'Flanagan, G. Paillard, R. Lavery, and A. M. Sengupta. Non-additivity in protein-DNA binding. *Bioinformatics*, 21(10):2254–2263, 2005.

G. Paillard and R. Lavery. Analyzing protein-DNA recognition mechanisms. *Structure*, 12: 113–122, 2004.

G. Paillard, C. Deremble, and R. Lavery. Looking into DNA recognition: Zinc finger binding specificity. *Nucleic Acids Research*, 32(22):6673–6682, 2004.

G. Pavesi, G. Mauri, and G. Pesole. An algorithm for finding signals of unknown length in DNA sequences. In *Proc. ISMB'01 (Supplement of Bioinformatics)*, pages 207–214, 2001.

G. Pavesi, G. Mauri, and G. Pesole. I*n silico* representation and discovery of transcription factor binding sites. *Briefings in Bioinformatics*, 5(3):217–236, 2004a.

G. Pavesi, P. Mereghetti, G. Mauri, and G. Pesole. Weeder Web: Discovery of transcription factor binding sites in a set of sequences from co-regulated genes. *Nucleic Acids Research*, 32(Web-Server-Issue):199–203, 2004b.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

S. V. Pemmaraju and S. S. Skiena. *Computational discrete mathematics: Combinatorics and graph theory with Mathematica.* Cambridge University Press, 2003.

F. Pernkopf and J. A. Bilmes. Discriminative versus generative parameter and structure learning of Bayesian network classifiers. In *Proc. ICML'05*, pages 657–664, 2005.

P. A. Pevzner and S. H. Sze. Combinatorial algorithm for finding subtle signals in DNA sequences. In *Proc. ISMB'00*, pages 269–278, 2000.

N. Pisanti, A. M. Carvalho, L. Marsan, and M.-F. Sagot. RISOTTO: Fast extraction of motifs with mismatches. In A. Hevia J. R. Correa and M. Kiwi, editors, *Proc. LATIN'06*, volume 3887 of *LNCS*, pages 757–768. 2006.

A. Policriti, N. Vitacolonna, M. Morgante, and A. Zuccolo. Structured motifs search. In *Proc. RECOMB'04*, pages 133–139, 2004.

J. V. Ponomarenko, M. P. Ponomarenko, A. S. Frolov, D. G. Vorobiev, G. C. Overton, and N. A. Kolchanov. Conformational and physicochemical DNA features specific for transcription factor binding sites. *Bioinformatics*, 15(7):654–668, 1999.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C: The art of scientific computing.* Cambridge University Press, 1993.

R. Pudimat, E. G. Schukat-Talamazzini, and R. Backofen. Feature based representation and detection of transcription factor binding sites. In *Proc. German Conference on Bioinformatics*, pages 43–52, 2004.

J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431, 1983.

J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, 1986.

J. Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society, Series B*, 49(3): 223–239, 1987.

J. Rissanen. *Stochastic Complexity in Statistical Inquiry Theory.* World Scientific, 1989.

J. Rissanen. Stochastic complexity and its applications. In *Workshop on Model Uncertainty and Model Robustness*, 1995a. On-line proceeding only.

J. Rissanen. Stochastic complexity in learning. In *Proc. EuroCOLT'95*, pages 196–210, 1995b.

J. Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, 1996.

T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri. On discriminative Bayesian network classifiers and logistic regression. *Machine Learning*, 59(3):267–296, 2005.

T. Roos, T. Silander, P. Kontkanen, and P. Myllymäki. Bayesian network structure learning using factorized NML universal models. In *Proc. ITA'08*, pages 272–276, 2008.

M.-F. Sagot. Spelling approximate repeated or common motifs using a suffix tree. In C. Lucchessi and A. Moura, editors, *Proc. Latin'98*, volume 1380 of *LNCS*, pages 111–127. 1998.

G. Sandve and F. Drablos. A survey of motif discovery methods in an integrated framework. *Biology Direct*, 1(1):11, 2006.

R. V. Satya and A. Mukherjee. New algorithms for finding monad patterns in DNA sequences. In Alberto Apostolico and Massimo Melucci, editors, *Proc. SPIRE'04*, volume 3246 of *LNCS*, pages 273–285. 2004.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

E. Segal, Y. Fondufe-Mittendorf, L. Chen, A. Thåström, Y. Field, I. K. Moore, J.-P. Z. Wang, and J. Widom. A genomic code for nucleosome positioning. *Nature*, 442(7104):772–778, 2006.

Y. M. Shtarkov. Universal sequential coding of single messages. *(Translated from) Problems of Information Transmission*, 23(3):3–17, 1997.

R. Siddharthan, E. D. Siggia, and E. van Nimwegen. PhyloGibbs: A Gibbs sampling motif finder that incorporates phylogeny. *PLoS Computational Biology*, 1(7):e67, 12 2005.

T. Silander, T. Roos, P. Kontkanen, and P. Myllymäki. Bayesian network structure learning using factorized NML universal models. In *Proc. PGM'08*, pages 257–264, 2008.

S. Sinha, M. Blanchette, and M. Tompa. PhyME: A probabilistic algorithm for finding motifs in sets of orthologous sequences. *BMC Bioinformatics*, 5:170, 2004.

G. D. Stormo. DNA binding sites: Representation and discovery. *Bioinformatics*, 16(1): 16–23, 2000.

J. Su and H. Zhang. Full Bayesian network classifiers. In *Proc. ICML'06*, pages 897–904, 2006.

J. Su, H. Zhang, C. X. Ling, and S. Matwin. Discriminative parameter learning for Bayesian networks. In *Proc ICML'08*, pages 1016–1023, 2008.

J. Suzuki. A construction of Bayesian networks from databases based on an MDL principle. In *Proc. UAI'93*, pages 266–273, 1993.

M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proc. UAI'05*, pages 584–591, 2005.

M. Tompa. An exact method for finding short motifs in sequences, with application to the ribosome binding site problem. In *Proc. ISMB'99*, pages 262–271, 1999.

M. Tompa, N. Li, T. L. Bailey, G. M. Church, B. De Moor, E. Eskin, A. V. Favorov, M. C. Frith, Y. Fu, W. J. Kent, V. J. Makeev, A. A. Mironov, W. S. Noble, G. Pavesi, G. Pesole, M. Regnier, N. Simonis, S. Sinha, G. Thijs, J. van Helden, M. Vandenbogaert, Z. Weng, C. Workman, C. Ye, and Z. Zhu. Assessing computational tools for the discovery of transcription factor binding sites. *Nature Biotechnology*, 23(1):137–144, 2005.

E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

A. Valouev, D. S. Johnson, A. Sundquist, C. Medina, E. Anton, S. Batzoglo, R. M. Myers, and A. Sidow. Genome-wide analysis of transcription factor binding sites based on ChiP-seq data. *Nature Methods*, 5:829–834, 2008.

J. van Helden, B. André, and J. Collado-Vides. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *Journal of Molecular Biology*, 281(5):827–842, 1998.

J. van Helden, A. F. Rios, and J. Collado-Vides. Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. *Nucleic Acids Research*, 28(8):1808–1818, 2000.

A. Vanet, L. Marsan, and M.-F. Sagot. Promoter sequences and algorithmical methods for identifying them. *Research in Microbiology*, 150(9–10):779–799, 1999.

A. Vanet, L. Marsan, A. Labigne, and M.-F. Sagot. Inferring regulatory elements from a whole genome. An analysis of H*elicobacter pylori* sigma(80) family of promoter signals. *Journal of Molecular Biology*, 297(2):335–353, 2000.

T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proc. UAI'90*, pages 255–270, 1990.

T. Wang and G. D. Stormo. Combining phylogenetic data with co-regulated genes to identify regulatory motifs. *Bioinformatics*, 19(18):2369–2380, 2003.

W. W. Wasserman and A. Sandelin. Applied bioinformatics for the identification of regulatory elements. *Nature reviews*, 5(4):276–287, 2004.

P. Weiner. Linear pattern matching algorithms. In *Proc. SWAT'73*, pages 1–11, 1973.

T. Werner. Models for prediction and recognition of eukaryotic promoters. *Mammalian Genome*, 10(2):168–175, 1999.

E. Wingender, X. Chen, E. Fricke, R. Geffers, R. Hehl, I. Liebich, M. Krull, V. Matys, H. Michael, R. Ohnhäuser, M. Prüss, F. Schacherer, S. Thiele, and S. Urbach. The TRANSFAC system on gene expression regulation. *Nucleic Acids Research*, 29(1):281–283, 2001.

E. P. Xing, M. I. Jordan, R. M. Karp, and S. J. Russell. A hierarchical Bayesian Markovian model for motifs in biopolymer sequences. In *Proc. NIPS'02*, pages 1489–1496, 2002.

S. Yang and K.-C. Chang. Comparison of score metrics for Bayesian network learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 32(3):419–428, 2002.

G. C. Yuan, Y. J. Liu, M. F. Dion, M. D. Slack, L. F. Wu, S. J. Altschuler, and O. J. Rando. Genome-scale identification of nucleosome positions in S*. cerevisiae*. *Science*, 309(5734): 626–630, 2005.

X. Zhao, H. Huang, and T. P. Speed. Finding short DNA motifs using permuted Markov models. In *Proc. RECOMB'04*, pages 68–75, 2004.

Q. Zhou and J. S. Liu. Modeling within-motif dependence for transcription factor binding site predictions. *Bioinformatics*, 20(6):909–916, 2004.