Hybrid learning of Bayesian multinet for binary classification

Alexandra M. Carvalho[1,4,⋆], Pedro Adão[2,5] and Paulo Mateus[3,5]

[1] *Department of Electrical Engineering, IST, University of Lisbon, Portugal*

[2] *Department of Computer Science, IST, University of Lisbon, Portugal*

[3] *Department of Mathematics, IST, University of Lisbon, Portugal*

[4] *PIA, Instituto de Telecomunicações, 1049-001 Lisbon, Portugal*

[5] *SQIG, Instituto de Telecomunicações, 1049-001 Lisbon, Portugal*

Abstract

We propose a scoring criterion, named *mixture-based factorized conditional log-likelihood* (mfCLL), which allows for efficient hybrid learning of mixtures of Bayesian networks in binary classification tasks. The learning procedure is decoupled in foreground and background learning, being the foreground the single concept of interest that we want to distinguish from a highly complex background. The overall procedure is hybrid as the foreground is discriminatively learned, whereas the background is generatively learned. The learning algorithm is shown to run in polynomial time for network structures such as trees and consistent κ-graphs. To gauge the performance of the mfCLL scoring criterion, we carry out an evaluation with state-of-the-art classifiers. Results obtained with a large suite of benchmark datasets show that mfCLL-trained classifiers are a competitive alternative and should be taken into consideration.

Key words: Conditional log-likelihood, approximation, hybrid learning, discriminative learning, Bayesian networks, mixtures, multinets.

1 Introduction

Bayesian networks [33] are probabilistic graphical models that represent the joint probability distribution of a set of random variables. They encode specific conditional independence properties pertaining to the joint distribution via a *directed acyclic graph* (DAG). To achieve this, each vertex (aka *node*) in the DAG contains a random variable, and edges between them represent the dependencies between the variables. Besides serving as a representation of a set of dependences, the DAG also allows to factorize the joint probability distribution via the chain rule of probability. The main advantage of Bayesian networks is that they can specify dependencies only when necessary, providing compact representations of complex domains which leads to a significant reduction in the cost of learning and inference.

Bayesian networks have been widely used for classification [18,21,40], being known in this context as *Bayesian network classifiers* (BNC). However, they are often outperformed by much simpler methods [15,18]. One of the most likely causes for this seems to be the use of so called *generative learning* methods in choosing the Bayesian network structure. In contrast to generative learning, where the goal is to be able to describe (or generate) the entire data, *discriminative learning* focuses on the capacity of a model to discriminate between different classes. Unfortunately, discriminative learning of BNCs turns out to be computationally much more challenging than generative learning. For this reason, the community has resorted to decomposing the learning procedure into generative-discriminative subtasks. In this context, Greiner and Zhou [20] proposed to generatively learn the network structure and to discriminatively learn the parameters, whereas in [21,41] the opposite is suggested. More recently, Carvalho *et al.* proposed an approximate approach for fully-discriminative learning of BNCs, exhibiting good performance both in terms of accuracy and computational cost [8].

A great effort has been done to understand the advantages and disadvantages of generative versus discriminative learning [35,31]. From this endeavor two regimes of performance, depending on the size of the training data, were elicited. For large datasets discriminative learning is preferred as it attains a lower asymptotic error. On the other hand, generative learning performs better with small datasets as it achieves its asymptotic error much faster. These empirical results motivated the development of hybrid approaches with intermediate regimes in between generative and discriminative limits, trying to get the best of both worlds. To this end, some approaches considered a convex combination of generative and discriminative likelihood functions [24,4]. Another work modified the naive Bayes model to learn a large subset of parameters based on the likelihood and a small subset of parameters based on the conditional likelihood [36]. In contrast, Bishop and Lasserre [3] blend generative and discriminative learning by introducing priors with constraints over the parameters which govern the balance between the two learning regimes.

Mixtures of Bayesian networks further generalize BNCs. Bayesian networks classifiers constrain the relations among the variables to be the same for all values of the class variable. A *mixture of Bayesian networks*, also called *Bayesian multinet* [19], can be thought as a Bayesian network where edges can appear and disappear depending on the values of certain nodes in the graph. In particular, Bayesian multinets as classifiers allow edges to appear and disappear depending on the value of the class variable [18]. This property is called *asymmetric independence assumption* [19]. Contrarily to BNCs, Bayesian multinets have only been learned generatively. In this paper we propose a decomposable scoring criterion for hybrid learning of multinets by extending the work of Carvalho *et al.* [8] and capitalizing on the results on generative, discriminative and hybrid learning [35,31].

The proposed learning procedure is decoupled in foreground and background learning. The *foreground* consists in the concept of interest, for instance, a disease under study from a set of other diseases and healthy patient data. The *background* consists of a possible miscellaneous of concepts, for instance, one or more diseases that are not under study and/or healthy patient data. In this context, we propose to learn the background generatively since generative learning achieves its asymptotic error faster, which is an advantage to learn highly complex data with a moderate size. On the other hand, the foreground is proposed to be learned discriminatively as discriminative learning achieves lower asymptotic error and the foreground has low-complexity, thus being accurately learned with little data. This is achieved through a new scoring criterion named *mixture-based factorized conditional log-likelihood* (mfCLL). The Bayesian multinet provides an extra benefit in this setup allowing us to describe the foreground and the background in two separate regimes with context-specific independences. The overall learning procedure is therefore hybrid, and it can be achieved in linear and polynomial time for network structures like trees [12,16] and consistent κ-graphs [6], respectively.

We applied the new hybrid procedure to distinguish *transcription factor binding sites* (TFBS) from non-binding DNA sequences. TFBSs are small strings of DNA where specific proteins bind to start the transcription of a gene. Transcription is the first step in gene expression and it is essential to protein biosynthesis which regulates the biochemical reactions inside the cell of living organisms. To evaluate the performance of mfCLL we assess its ability in representing TFBSs from 89 benchmark datasets used in previous works [1,7]. We first provide a comparison with existing TFBS models and then proceed to a broad examination with state-of-the-art classifiers, namely, *support vector machine* (SVM), *logistic regression* (LogR), *decision tree* (DT), *k-nearest neighbor* (*k*-NN) and *tree augmented naive Bayes with factorized conditional log-likelihood* (TAN-fCLL) classifiers. On the above mentioned 89 datasets, a specific mfCLL-based mixture provided a higher discriminative power than previous TFBS models. Moreover, such mfCLL-trained mixture outperformed, with high significance level, SVMs with linear and Gaussian kernels, LogR, DT, 3-NN, 5-NN, and TAN-fCLL classifiers. In addition, it showed to be comparable with polynomial SVMs. Notwithstanding, mfCLL-based mixtures are computationally more efficient than polynomial SVMs, taking 2 to 3 orders of magnitude less time for the 89 considered datasets.

The elicited mfCLL-trained mixture was also evaluated over eight UCI diagnosis datasets [30]. The results confirmed its superior power since it outperformed with statistical significance all other classifiers.

The paper is organized as follows. In Section 2 we review the essentials of Bayesian networks, as well as generative, discriminative and hybrid learning. In Section 3 we start by providing background material in Bayesian multinets and proceed to present our scoring criterion for discriminatively learning the mixture foreground. Given the model selection criterion we derive the optimal parameters and present the hybrid procedure to learn the mixture structure. In Section 4 we assess the developed techniques against state-of-the-art classifiers. Finally, we draw some conclusions and future work in Section 5.

2 Background

In this section we review the basic concepts of Bayesian networks required to understand the proposed methods, and discuss the differences between generative and discriminative learning of BNCs.

2.1 Bayesian networks

Let X be a *discrete random variable* taking values in a countable set $\mathcal{X}$. In all that 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 that X_i may 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 $P(\mathbf{x})$, being conditional probabilities $P(\mathbf{x} \mid \mathbf{z})$ defined accordingly. The random vector $\mathbf{X}$ is said to be *conditionally independent* of random vector $\mathbf{Y}$ given random vector $\mathbf{Z}$ if $P(\mathbf{x} \mid \mathbf{y}, \mathbf{z}) = P(\mathbf{x} \mid \mathbf{z})$, for all $\mathbf{x}, \mathbf{y}$, and $\mathbf{z}$.

A *Bayesian network* (BN) is a triple $B = (\mathbf{X}, G, \Theta)$ where $\mathbf{X} = (X_1, \ldots, X_n)$ is a random vector. The network structure $G = (\mathbf{X}, E)$ is a *directed acyclic graph* (DAG) with nodes in $\mathbf{X}$ and edges E representing direct dependencies between the variables. 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, we denote by Π_{X_i} the (possibly empty) set of *parents* of X_i in G. For each node X_i, the number of possible vectors of parents' values, called the *parent configurations*, is denoted by q_i. The actual parent configurations are ordered (arbitrarily) and denoted by $w_{i1}, \ldots, w_{iq_i}$. For $j \in \{1, \ldots, q_i\}$, we say that w_{ij} is the j-th configuration of Π_{X_i}. Using this notation, the third element of the BN triple denotes the *parameters* $\Theta = \{\theta_{ijk}\}_{i \in \{1 \ldots n\}, \, j \in \{1, \ldots, q_i\}, \, k \in \{1, \ldots, r_i\}}$ that encode the *local distributions* of the network via

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

A BN B induces a joint probability distribution over $\mathbf{X}$ given by

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

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 [33].

The problem of learning a BN, given some data, consists of finding the BN that best fits the underlying distribution generating the data. This can be achieved by a score-based learning algorithm, where a *scoring criterion* is considered in order to quantify the fitting of a BN.

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*, that is, if the scoring criterion can be expressed as a sum of local scores associated to each network node and its parents. In this case, whenever the network structure changes during the search procedure, the score of the new network is re-evaluated just by modifying the local contribution of the changed part.

The most common scoring criteria employed in BN learning are reviewed in [5,44,23]. We refer the interested reader in newly developed scoring criteria to the works of Carvalho *et al.* [8], de Campos [13] and Silander *et al.* [39].

2.2 Generative, discriminative and hybrid learning of Bayesian networks

In classification tasks BNs are used as Bayesian network classifiers. Rigorously, a *Bayesian network classifier* (BNC) is a BN 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*. 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. Rigorously, an *augmented naive Bayes classifier* is a BNC 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, $C \in \Pi_{X_i}$ for all $1 \leq i \leq n$. In this work we focus our attention on augmented naive Bayes classifiers, referring abusively to them as BNCs.

For convenience, we introduce additional notation. Let data T with size N be given by $T = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, where: (i) $\mathbf{x}_t = (x_t^1, \ldots, x_t^n, c_t)$ denotes the t-th instance of the data; (ii) $x_t^1, \ldots, x_t^n$ are the values that the attributes $X_1, \ldots, X_n$ take in the t-th instance; and (iii) c_t is the corresponding value for the class variable C.

Generative learning aims at maximizing the likelihood of the data, by using the log-likelihood scoring criterion or a score thereof (for instance, [39,13]). The *log-likelihood* scoring criterion can be written as:

$$\mathrm{LL}(B \mid T) = \sum_{t=1}^{N} \log(P_B(x_t^1, \ldots, x_t^n, c_t)). \tag{2}$$

Discriminative learning, on the other hand, aims at maximizing the conditional likelihood of the data. The reason why this is a form of discriminative learning is that it focuses 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:

$$\mathrm{CLL}(B \mid T) = \sum_{t=1}^{N} \log(P_B(c_t | x_t^1, \ldots, x_t^n)). \tag{3}$$

Friedman *et al.* [18] noticed that, in the context of classification learning problems, the log-likelihood of T for B can be rewritten as:

$$\mathrm{LL}(B \mid T) = \mathrm{CLL}(B \mid T) + \sum_{t=1}^{N} \log(P_B(x_t^1, \ldots, x_t^n)). \tag{4}$$

Interestingly, the objective of generative learning is precisely to maximize the whole sum, whereas the goal of discriminative learning consists of maximizing only the first term of the sum in (4). Friedman *et al.* [18] attributed the underperformance of learning methods based on LL to the term $\mathrm{CLL}(B \mid T)$ being potentially much smaller than the second term in (4). Unfortunately, CLL does not decompose over the network structure, and therefore there is no closed-form equation for optimal parameter estimates for the CLL scoring criterion. The first works in this line of research split the problem into two distinct generative-discriminative tasks: (i) find optimal-CLL parameters and optimal LL-structure [20,41] and; (ii) find optimal-CLL structure and optimal LL-parameters [2,21]. Although showing promising results, these hybrid approaches present a problem of computational nature. Indeed, optimal-CLL parameters have been achieved by resorting to gradient descent methods, and optimal-CLL structures have been found only with global search methods, making both methods very inefficient. Recently, a least-squares approximation to CLL that enables full discriminative learning of BNCs in a very efficient way has been proposed [8]. This paper extends further this work dealing with mixtures of BNs where the components of the mixtures are learned with distinct regimes.

3 Hybrid learning of a two-component multinet

Herein, we present the concepts related to two-component mixtures of BNs and motivate the hybrid procedure to learn them. Then, we establish our decomposable scoring criterion for learning the mixture foreground. Finally, we provide methods for parameter and structure learning based on the proposed model selection criterion.

3.1 Two-component multinets

Probabilistic mixtures of general graphical models were introduced by Geiger and Heckerman [19] and since then they have been utterly applied in several domains [29,18]. Mixtures of arbitrary graphical models are also called *Bayesian multinets* (BM). The main advantage of BMs is that they allow to represent *context-specific independences*. We find these 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 some additional notation for BMs intended to be learned from data $T = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, where $\mathbf{x}_t = (x_t^1, \dots, x_t^n, c_t)$. The last component of each instance in T is the value of the class variable C, that is, c_t is the value of the class variable C in the t-th instance of T. In binary classification tasks, this variable ranges over the set $\mathcal{C} = \{0, 1\}$. It is also useful to associate to each index of an instance of T its corresponding class value. More precisely, consider the map $\eta : \{1, \dots, N\} \to \mathcal{C}$, where $\eta(t) = c_t$. Moreover, for $\mathbf{x}_t = (x_t^1, \dots, x_t^n, c_t)$ and $c \in \{0, 1\}$ we set the following notation:

$$\mathbf{x}_t^- = (x_t^1, \dots, x_t^n), \quad I_c = \eta^{-1}(c) \quad \text{and} \quad T_c = \{\mathbf{x}_t^- : t \in I_c\}.$$

Loosely speaking, I_c is the set of indexes of the instances in T where the class variable takes the value c and T_c is the set of instances, excluding the class variable, for which the class variable takes the value c.

A *two-component Bayesian multinet*, or *two-component mixture of Bayesian networks*, is a triple $\mathcal{M} = \langle \{\lambda_c\}_{c=0,1}, B_0, B_1 \rangle$ where $\lambda_c = P(C = c)$ is called the *mixing proportion* and each B_c is a BN over $\{X_1, \dots, X_n\}$. The BN for each value c is called the *local Bayesian network* for c. For the sake of simplicity, we omit λ_1 from the two-component mixture model $\mathcal{M}$ as $\lambda_1 = 1 - \lambda_0$, referring abusively to it as

$$\mathcal{M} = \langle \lambda_0, B_0, B_1 \rangle.$$

Without loss of generality, we henceforth call class 0 the *background* and class 1 the *foreground*; that is, B_0 is the mixture background model and B_1 is the mixture foreground model.

A BM defines a unique joint probability distribution given by:

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

The standard procedure to learn BMs is to compute from data

$$\hat{\lambda}_c = \hat{P}_T(C = c) = \frac{N_c}{N}, \tag{5}$$

where N_c is the number of instances in the data T where the class variable C takes the value c and $\hat{P}_T$ is the distribution induced by the *observed frequency estimates* (OFE). Each local BN B_c is then learned independently over the subset T_c. Predictions are made by choosing the class variable that maximizes the posterior probability $P_{\mathcal{M}}(C \mid X_1, \ldots, X_n)$.

3.2 Learning approach

Before establishing the discriminative scoring criterion for learning the foreground of a two-component BM we motivate the proposed learning approach.

As mentioned before, the envisaged learning procedure will be decoupled in foreground and background learning. The rationale for this approach is that we want to distinguish some particular concept (the foreground) from a miscellaneous of other concepts forming the background. This is common in many real-word applications. For instance, in medicine, a physician may want to distinguish a given disease (under study) from a set of other diseases and/or healthy patient data. Several other examples arise from computational molecular biology where, for instance, detecting special signals from the DNA sequence is of the utmost importance. A DNA sequence comprises both coding and non-coding regions, possibly containing special regulation signals and noise, respectively. In this context, one may want to elicit a certain signal and distinguish it from the remaining background signals and/or noise (see Section 4.1 for a detailed example in this setting).

Although the envisaged hybrid approach is intuitive *per se*, there is also a strong motivation for decoupling the learning in a generative-discriminative procedure. Undoubtedly, understanding the pros and cons of generative and discriminative learning is crucial to devise hybrid approaches that enjoy the best properties of both worlds [31]. In our framework, background data is usually noisy, highly-complex and may have multiple and overlapping concepts. Generative learning seems to deal with this type of data better as it approaches its asymptotic error much faster than discriminative learning. In order to be able to use discriminative methods to learn the background the amount of data needed would be much larger, posing two different problems: (i) this data may not be available; (ii) the learning might become imbalanced producing classifiers with a huge bias to the majority class. On the other hand, since the foreground class is usually less complex with very well defined patterns, a small amount of data is enough for accurate predictions. As discriminative methods have lower asymptotic error than generative ones [31] they seem more suitable for foreground learning.

Before proceeding, it is worthwhile pointing out the differences between pure generative and the proposed hybrid learning of BMs. In pure generative learning each local BN B_c is learned using only the data T_c (disregarding T_{1-c}) [19]. In the proposed hybrid procedure, B_0 is learned generatively whereas B_1 is learned discriminatively. Thus, while T_0 is sufficient to learn B_0, both T_0 and T_1 are required to learn B_1. Indeed, in order to distinguish the foreground from the background, information from both classes is needed and therefore both the observations from T_0 and T_1 are relevant. In what follows, we will provide a discriminative scoring criterion to learn B_1 from the full dataset T. We henceforth assume that B_0 was already learned with some generative learning procedure in the literature and that λ_0 and λ_1 were computed as in (5).

3.3 Model selection

Herein, we present the new discriminative scoring criterion to learn the foreground of a two-component BM. This new score will be used to select the foreground model that best fits the (conditional) distribution underlying the data.

It is convenient to extend the notation introduced in the previous sections to cope with the discriminative learning of B_1 within a mixture $\mathcal{M} = \langle \lambda_0, B_0, B_1 \rangle$. Henceforth, the usage of the superscript 1 means that we are referring to the BN B_1 only. 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 similarly 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. Finally, θ^1_{ijk} denotes the probability $P_{B_1}(X_i = x_{ik} \mid \Pi^1_{X_i} = w^1_{ij})$, representing the parameters of B_1.

In the context of a BM $\mathcal{M} = \langle \lambda_0, B_0, B_1 \rangle$ for binary classification tasks, we have that

$$P_{\mathcal{M}}(c_t \mid y^1_t, \ldots, y^n_t) = \frac{\lambda_{c_t} P_{B_{c_t}}(y^1_t, \ldots, y^n_t)}{\lambda_{c_t} P_{B_{c_t}}(y^1_t, \ldots, y^n_t) + \lambda_{(1-c_t)} P_{B_{(1-c_t)}}(y^1_t, \ldots, y^n_t)}. \quad (6)$$

To simplify notation, let

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

hence, expression (6) 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. 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. Therefore, in order to decompose $\mathrm{CLL}(\mathcal{M} \mid T)$ over the two-component network structures, the function

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

needs to be approximated by

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

where U_t and V_t are probabilities.

To analytically obtain the real numbers α, β and γ that best approximate $\hat{f}$ to f we need to make some reasonable stochastic assumptions about U_t and V_t, even if they do not hold true exactly. As U_t and V_t correspond to joint probabilities that are very small, we follow the reasoning of Carvalho *et al.* [8] and assume that $(U_t, V_t) \sim \mathrm{Unif}([0, p]^2)$, for some small probability p. In this case, the values for the constants α, β and γ that minimize the mean square error were computed analytically in [8] and are given by:

$$\alpha = \frac{\pi^2 + 6}{24}, \quad \beta = \frac{\pi^2 - 18}{24} \quad \text{and} \quad \gamma = \frac{\pi^2}{12\ln(2)} - \left(2 + \frac{(\pi^2 - 6)\log(p)}{12}\right).$$

This results in the following decomposable approximation for the CLL

$$\mathrm{CLL}(\mathcal{M} \mid T) = \sum_{t=1}^{N} \log\left(\frac{U_t}{U_t + V_t}\right) \approx \sum_{t=1}^{N} \left(\alpha \log(U_t) + \beta \log(V_t) + \gamma\right). \tag{7}$$

In [8] it is shown that this approximation is unbiased, i.e., $E[\hat{f}(U_t, V_t) - f(U_t, V_t)] = 0$, and that it minimizes the variance, i.e., $E[(\hat{f}(U_t, V_t) - f(U_t, V_t))^2]$ is minimal. As we shall see later in this paper, the dependence of γ on p is irrelevant as γ will be included in a constant that will be discarded, since it has no role in model selection.

As the sum over t in (7) is ranging over all the dataset T we can decouple the sum in two parts, one accounting for the contribution of T_0 and another for the contribution of T_1. In this way, we have that

$$\mathrm{CLL}(\mathcal{M} \mid T) = \mathrm{CLL}(\mathcal{M} \mid T_0) + \mathrm{CLL}(\mathcal{M} \mid T_1).$$

Moreover, assuming that both the mixing proportion λ_0 and the background model B_0 were (previously) generatively learned (and so are fixed), and knowing that $\lambda_1 = 1 - \lambda_0$, we only need to learn the foreground model B_1. In this case, we have that

$$\begin{aligned}
\mathrm{CLL}(B_1 \mid T_0) &\approx \sum_{t \in T_0} \left(\alpha \log(U_t) + \beta \log(V_t) + \gamma\right) \\
&= \sum_{t \in I_0} \Big(\alpha \log(\lambda_0 P_{B_0}(\mathbf{x}_t^-)) + \beta \log(\lambda_1 P_{B_1}(\mathbf{x}_t^-)) + \gamma\Big) \\
&= \left(\sum_{t \in I_0} \beta \log(P_{B_1}(\mathbf{x}_t^-))\right) + \sum_{t \in I_0} \Big(\alpha \log(\lambda_0 P_{B_0}(\mathbf{x}_t^-)) + \beta \log(\lambda_1) + \gamma\Big) \\
&= \left(\sum_{t \in I_0} \beta \log(P_{B_1}(\mathbf{x}_t^-))\right) + K_0,
\end{aligned}$$

where K_0 accounts for the (fixed) contribution of $B_0, \lambda_0, \lambda_1$ and γ to $\mathrm{CLL}(B_1 \mid T_0)$. The notation with B_1 as argument instead of $\mathcal{M} = \langle \lambda_0, B_0, B_1 \rangle$ in CLL emphasizes that the criterion is a function of the foreground model B_1 only, since B_0, λ_0 and λ_1 are fixed. Similarly,

$$
\begin{aligned}
\mathrm{CLL}(B_1 \mid T_1) &\approx \sum_{t\in T_1} (\alpha \log(U_t) + \beta \log(V_t) + \gamma) \\
&= \sum_{t\in I_1} \Big(\alpha \log(\lambda_1 P_{B_1}(\mathbf{x}_t^-)) + \beta \log(\lambda_0 P_{B_0}(\mathbf{x}_t^-)) + \gamma\Big) \\
&= \left(\sum_{t\in I_1} \alpha \log(P_{B_1}(\mathbf{x}_t^-))\right) + \sum_{t\in I_1} \Big(\alpha \log(\lambda_1) + \beta \log(\lambda_0 P_{B_0}(\mathbf{x}_t^-)) + \gamma\Big) \\
&= \left(\sum_{t\in I_1} \alpha \log(P_{B_1}(\mathbf{x}_t^-))\right) + K_1,
\end{aligned}
$$

where K_1 accounts for the (fixed) contribution of $B_0, \lambda_0, \lambda_1$ and γ to $\mathrm{CLL}(B_1 \mid T_1)$. Since the constants K_0 and K_1 are irrelevant for maximizing $\mathrm{CLL}(B_1 \mid T_0)$ and $\mathrm{CLL}(B_1 \mid T_1)$, respectively, we can drop them. Therefore, we define the *mixture-based factorized conditional log-likelihood* (mfCLL) scoring criterion as

$$
\begin{aligned}
\mathrm{mfCLL}(B_1 \mid T) &= \left(\sum_{t\in I_0} \beta \log(P_{B_1}(\mathbf{x}_t^-))\right) + \left(\sum_{t\in I_1} \alpha \log(P_{B_1}(\mathbf{x}_t^-))\right) \\
&= \left(\sum_{i=1}^{n}\sum_{j=1}^{q_i^1}\sum_{k=1}^{r_i} \beta N_{ij0k}^1 \log(\theta_{ijk}^1)\right) + \left(\sum_{i=1}^{n}\sum_{j=1}^{q_i^1}\sum_{k=1}^{r_i} \alpha N_{ij1k}^1 \log(\theta_{ijk}^1)\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{q_i^1}\sum_{k=1}^{r_i} (\alpha N_{ij1k}^1 + \beta N_{ij0k}^1) \log\left(\theta_{ijk}^1\right). \qquad (8)
\end{aligned}
$$

The choice of the scoring criterion to use is particularly relevant when the complexity of the BN structure grows. For instance, the LL scoring criterion in (2) tends to favor complete network structures since adding an edge never decreases the likelihood on the training data. This phenomenon leads to *overfitting* which is usually avoided by adding a complexity penalty to the LL. Most common penalties are information-theoretic and so based on compression. In this case, the score of a BN B is related to the compression that can be achieved over the data 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 (usually measured in bits).

Rissanen proposed to assume that only integers are used to encode the parameters of a BN B and to use the optimal *(universal) code for integer* to encode them [37,38]. In this case, the number of bits required to represent B is

$$K(B) = \frac{1}{2}\log(N)|B|,$$

where $|B|$ is the total number of parameters of B. This approach led to the development of the *minimum description length* (MDL) scoring criterion defined as:

$$\mathrm{MDL}(B \mid T) = \mathrm{LL}(B \mid T) - K(B).$$

The MDL criterion has also been used in the context of multinets [18] and discriminative learning [26]. For multinets, the MDL penalty of $\mathcal{M}$ is given by

$$K(\mathcal{M}) = \sum_c K(B_c) + \frac{1}{2}\log(N)(|\mathcal{C}| - 1),$$

which is the sum of the MDL penalties of each local BN B_c jointly with the multinomial for the class. Usually, the MDL penalty of the multinomial of the class is not considered as it is irrelevant for maximizing the score (because it is constant given the dataset). As in generative learning of multinets, in our hybrid procedure the (generative) background B_0 does not depend on the foreground network B_1, and the (discriminative) foreground B_1 considers the background network B_0 fixed. Therefore, a MDL penalized version of the mfCLL score should only take into account the foreground structure B_1. Such penalized version, called mfCLL$^{\mathrm{MDL}}$, can be straightforwardly obtained by subtracting the MDL penalty to (8) resulting in the following decomposable score:

$$\mathrm{mfCLL}^{\mathrm{MDL}}(B_1 \mid T) = \mathrm{mfCLL}(B_1 \mid T) - \frac{1}{2}\ln(N)\sum_{i=1}^{n}(r_i - 1) \times q_i^1. \qquad (9)$$

3.4 Parameter learning

It this section we derive the values of the parameters θ^1_{ijk} that maximize mfCLL in (8); the same parameters maximize (9). We are able to obtain the optimal values of θ^1_{ijk} by assuming that they are lower-bounded. This lower bound on θ^1_{ijk} follows from adopting pseudo-counts, commonly used in BNCs and BMs to smooth observed frequencies with Dirichlet priors and increase the quality of the classifier [18]. Pseudo-counts attend to impose the common sense assumption that there are no situations with probability zero. Indeed, it is a common mistake to assign probability zero to an event that is extremely unlikely, but not impossible [28].

Theorem 3.1 Let $N' > 0$ be the number of pseudo-counts. The parameters θ^1_{ijk} that maximize (8) are given by

$$\theta^1_{ijk} = \frac{N^1_{ij+k}}{N^1_{ij+}} \tag{10}$$

where

$$N^1_{ij+k} = \begin{cases} \alpha N^1_{ij1k} + \beta N^1_{ij0k} & \text{if } \alpha N^1_{ij1k} + \beta N^1_{ij0k} \geq N' \\ N' & \text{otherwise} \end{cases}$$

and

$$N^1_{ij+} = \sum_{k=1}^{r_i} N^1_{ij+k},$$

constrained to $\theta^1_{ijk} \geq \frac{N'}{N^1_{ij+}}$ for all i, j and k.

Proof: Note that

$$\begin{aligned} \text{mfCLL}(B_1 \mid T) &= \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} (\alpha N^1_{ij1k} + \beta N^1_{ij0k}) \log\left(\theta^1_{ijk}\right) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \left(\underbrace{N^1_{ij+k} \log\left(\theta^1_{ijk}\right)}_{(a)} + \underbrace{((\alpha N^1_{ij1k} + \beta N^1_{ij0k}) - N^1_{ij+k}) \log\left(\theta^1_{ijk}\right)}_{(b)} \right). \end{aligned} \tag{11}$$

Observe that if $N^1_{ij+k} \geq N'$ then $N^1_{ij+k} = (\alpha N^1_{ij1k} + \beta N^1_{ij0k})$. Thus the summand (b) in (11) is only different from zero when $(\alpha N^1_{ij1k} + \beta N^1_{ij0k}) < N'$. In this case $N^1_{ij+k} = N'$ which implies that

$$(\alpha N^1_{ij1k} + \beta N^1_{ij0k}) - N^1_{ij+k} < 0.$$

So, the value for θ^1_{ijk} that maximizes the summand (b) is the minimal value for θ^1_{ijk}, that is, $\frac{N'}{N^1_{ij+}} = \frac{N^1_{ij+k}}{N^1_{ij+}}$. Finally, by Gibb's inequality, we derive that the distribution for θ^1_{ijk} that maximizes the summand (a) in (11) is $\theta^1_{ijk} = \frac{N^1_{ij+k}}{N^1_{ij+}}$. Since the maximality of the summands (a) and (b) is obtained with the same distribution, we have that the values for θ^1_{ijk} that maximize (8) are given by $\theta^1_{ijk} = \frac{N^1_{ij+k}}{N^1_{ij+}}$. $\square$

The role of the pseudo-counts N' is to guarantee that no counting N^1_{ij+k} can be below it. By plugging the parameters obtained in (10) into the mfCLL criterion in (8), we obtain

$$\widehat{\text{mfCLL}}(G_1 \mid T) = \sum_{i=1}^{n}\sum_{j=1}^{q_i^1}\sum_{k=1}^{r_i}(\alpha N^1_{ij1k} + \beta N^1_{ij0k}) \log\left(\frac{N^1_{ij+k}}{N^1_{ij+}}\right). \tag{12}$$

The notation with G_1 has argument instead of $B_1 = (\mathbf{X}, G_1, \Theta_1)$ emphasizes that once the parameters Θ_1 are decided upon, the criterion is a function of the network structure G_1 only. The MDL penalized version of (12) is given straightforwardly by

$$\widehat{\text{mfCLL}}^{\text{MDL}}(G_1 \mid T) = \widehat{\text{mfCLL}}(G_1 \mid T) - \frac{1}{2}\ln(N)\sum_{i=1}^{n}(r_i - 1) \times q_i^1. \tag{13}$$

Next, we show that mfCLL in (12) is not score equivalent; and consequently the same applies for (13). Two BNs are said to be *equivalent* if they can represent precisely the same set of distributions. Verma and Pearl [42] showed that this is equivalent to check if the underlying DAGs of the two BNs have the same skeleton and the same v-structures. A *score-equivalent* scoring criterion is one that assigns the same score to equivalent BN structures [10,44,13].

Theorem 3.2 The $\widehat{\text{mfCLL}}$ scoring criterion is decomposable and non-score equivalent.

Proof: Decomposability follows directly from the definition in (12). Concerning non-score equivalence, it suffices to provide a counter-example where two equivalent structures do not score the same. To this purpose consider a two-node multinet ($n = 2$) and $T = \{(0,0,1),(0,1,1),(1,1,0),(1,1,1)\}$ as the training set. The structures $G \equiv X_1 \to X_2$ and $H \equiv X_2 \to X_1$ for B_1 are equivalent, but it is easy to check that $\widehat{\mathrm{mfCLL}}(G \mid T) \neq \widehat{\mathrm{mfCLL}}(H \mid T)$. $\square$

Since learning using undecomposable scores is, in general, an expensive task, most of the interesting scoring criteria in the literature are decomposable. It is however possible to employ undecomposable scores avoiding this computational expensiveness considering prior assumptions on the possible network structures [9]. In addition, both score-equivalent and non-score-equivalent decomposable scores can be learned efficiently, although the algorithms to learn them are different. Non-score-equivalent scores typically perform better than score-equivalent ones [13,44].

3.5 Structure learning

In this section we briefly discuss two well-known algorithms for local score-based learning. The algorithm for hybrid learning of the foreground and the background is then presented, profiting from these discussions.

For decomposable scores, optimal networks can be learned locally being the global score of the network taken as a sum or a product of the local contributions of the score. Notwithstanding, local score-based learning is efficient only if restricted to certain structures, like trees [12,16] and consistent κ-graphs (CκG) [6]. Unrestricted BN structure learning is known to be NP-hard [11], even for decomposable scores. Thus, to achieve efficient hybrid learning of BMs we restrict our attention to trees and CκG networks.

Tree BNs are learned using the Chow-Liu's [12] or Edmonds' [16] algorithms. The first is used with score-equivalent scoring criteria and the latter with non score-equivalent ones. The Chow-Liu's algorithm simply finds the maximum spanning tree from a complete weighted graph. Edmonds' algorithm finds the maximum directed spanning tree from a directed weighted graph, where edges from A to B and B to A might weight differently. In both algorithms, each edge of the complete graph is weighted with the local contribution of the edge for the global score. Because of this, only decomposable scores can be applied. The optimal tree is found in $O(Nn^2)$ time for both Chow-Liu's and Edmonds' algorithms.

Although very efficient, tree networks significantly restrict the possible dependencies between variables since it only allows one parent per node. Consistent κ-graphs overcome this shortcoming, allowing for dependencies that are consistent with a topological order of the optimal tree [6]. In detail, after obtaining the optimal tree the breadth-first search (BFS) order of this tree is considered. Then, each node may have at most κ parents, provided that the parents are smaller than the node in the BFS order. For a fixed κ, CκG networks can be learned in polynomial-time. Moreover, the optimal CκG network always scores better than the optimal tree.

We are now able to present the algorithm for learning two-component mixtures of BNs for binary classification tasks. The learning procedure, presented in Algorithm 1, relies on two other algorithms that learn each mixture component B_0 and B_1. It starts by computing the mixing proportions $\lambda = \langle\lambda_0, \lambda_1\rangle$ as in (5). Then, it generatively learns the background model B_0, and discriminatively learns the foreground model B_1. The overall procedure is therefore hybrid.

Algorithm 1 Hybrid learning of two-component multinets

```
(1) Compute the mixing proportions $\lambda_0 = \frac{N_0}{N}$ and $\lambda_1 = 1 - \lambda_0$.
(2) Learn generatively from $T_0$ the BN $B_0$.
(3) Learn discriminatively from $T$, with the $\widehat{\mathrm{mfCLL}}$ score or a score thereof, the BN $B_1$.
```

It is important to notice that any BN learning algorithm that copes with decomposable scores can be used in Step 3 of Algorithm 1. For instance, Chow-Liu's [12] or Edmonds' [16] algorithms can be used to learn tree-like structures. Moreover, CκG network structures [6] can also be learned. Although trees and CκGs are usually good candidates, as they offer efficient and optimal solutions, heuristic algorithms like greedy Hill climber can also be employed. The only premise is that the algorithm should be based on local score optimization, in order to be able to employ in Step 3 the mfCLL scoring criterion in (12), or a penalized version of it as in (13). Concerning Step 2, only T_0 is considered to learn the background model B_0 and any generative learning algorithm can be employed.

The time complexity of Algorithm 1 only depends on the time complexity of the algorithms employed to learn B_0 and B_1 in steps 2 and 3. For instance, if tree-like structures are learned in both steps then the resulting algorithm is linear in the size of the data T, that is, $O(Nn^2)$. However, if CκG-like structures are learned in any (or both) of the steps then the algorithm is polynomial in κ, that is, $O(Nn^{\kappa+1})$.

4 Experimental results

In this section we present the experimental methodology along with the results and their interpretation. We start by presenting a particular setting in the field of computational biology where discriminative learning of mixtures of BNs plays an important role. We then compare our results with previous works in this same core, mainly focusing in BN-based models. Moreover, we perform a deep comparison with state-of-the-art classifiers. The best mixture elicited from these results is then evaluated with UCI medical diagnosis datasets.

We implemented the CκG-mixture model and mfCLL scoring criterion in Mathematica 7.0 on top of the Combinatorica package [34]; plain CκG models were already implemented on top of this package by the authors [7]. Both plain and penalized versions of mfCLL, given by equations (12) and (13), were implemented. In our experiments we used $\kappa = 2$ since it showed to be a good tradeoff between efficiency and expressiveness [6].

4.1 Modeling transcription factor binding sites

An important part of gene regulation is mediated by specific proteins, called *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). Such binding sites are relatively short strings of DNA, normally 5 to 25 nucleotides long. A nucleotide is a letter of the DNA alphabet, given by $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. Usually, these binding sites are moderately conserved strings so that they are recognized by the TF as sites to bind and start transcription of a particular gene.

When dealing with the TFBS representation one needs a collection of known binding sites, which in practice constitutes a collection of moderately conserved DNA substrings. These strings are then aligned and trimmed to have the same size. The size of the binding sites corresponds to the number of features of the probabilistic models used to represent them. Each feature is nominal and ranges over by the DNA alphabet. TFBS models are ultimately needed to discriminate binding regions from non-binding ones. This is a classification task. To this end, the collection of known binding sites are labelled as “binding”, and joined to a collection of non-binding sites that are labelled as “non-binding”, and a classification task is performed. As DNA is a single sequence, “non-binding” instances correspond to subsequences of DNA that are not TFBSs. These non-binding sequences are usually retrieved from surrounding regions of TFBSs, typically from upstream regions of genes, and then trimmed to have the same size as binding instances.

A commonly used representation of TFBSs is a *position-specific scoring matrix* (PSSM). It is defined by a matrix where each entry (i, b) is the probability of nucleotide b being at the i-th position in the collection of binding sites. This representation assumes independence of nucleotides in the binding sites. Although popular, this simplistic independence assumption paved the way for more complex models that account for nucleotide interactions [32]. Barash *et al.* [1] already obtained good results modeling TFBSs with generatively learned tree BNs and mixtures of trees. More recently Carvalho *et al.* [7] also contributed in this direction by proposing pure generative C2G models to represent TFBSs. Herein, we evaluate the extent to which the hybrid C2G-mixture models are beneficial in representing TFBSs.

In TFBS representation there is no reason to think that a unique BN is suitable to represent the co-regulated DNA sequences (the background) and, at the same time, a TFBS within such region (the foreground). Indeed, the background contains several signals along with noise, as it might contain both coding and non-coding DNA regions, whereas the foreground contains a collection of binding sites moderately conserved. These two separate regimes are, almost certainly, the reason why tree-mixture models have shown to be better suited than plain tree models for TFBS representation [1]. When compared with tree mixtures [1], CκG-mixture models allow important nucleotide interactions to be captured. In this context, CκG models allow for κ dependencies between nucleotides whereas tree models allow only for one interaction (and PSSM models for none). Due to this fact, CκG allows for v-structures in its underlying DAG and so exhibits the so called *induced dependencies*, where totally unrelated propositions become relevant to each other when new facts are learned. In the case of TFBSs these dependencies may capture, for instance, that the first position of a TFBS is independent from the second one, unless in the third position there is a certain nucleotide. PSSM and tree models are unable to provide such dependencies.

4.1.1 Evaluating Bayesian network-based models

We evaluated the performance of hybrid C2G-models in TFBS representation comparing them with state-of-the-art BN-based models. We performed our evaluation on the same 89 benchmark datasets used by Barash *et al.* [1]. These datasets were taken from the TRANSFAC database [43], containing hundreds of biologically validated TFBSs.

These 89 sequence-sets, constituting each one a foreground data T_1, were extracted from aligned binding sites of *Saccharomyces cerevisiae* for which there were 20 or more sites. Therefore, the minimum number of instances in T_1 is 20, whereas the largest T_1 contains 80 instances. Each feature in T_1 corresponds to a specific position of the binding sites (first feature is the first position, second feature is the second position, and so on). In each of these positions there is a letter of the DNA alphabet, so features are nominal and contain the letters `A`, `C`, `G` or `T`. The number of features in the 89 sequence-sets ranges from 6 to 24, corresponding to the size of the binding sites.

The background data T_0 was gathered from the upstream regions of genes of the same organism (in this case, *Saccharomyces cerevisiae*). This corresponds to non-binding sites in the DNA. In practice, 1000 non-binding sequences were collected for each dataset T_0. The goal of classification is to distinguish binding sites in T_1 from non-binding sites in T_0. The resulting 89 datasets comprise both the foreground and the background data. Moreover, in each of these 89 datasets a new additional feature was added representing the respective class variable, that is, instances in T_0 were labelled with 0, whereas instances in T_1 were labelled with 1. Thus, the number of instances of the final 89 datasets varies from 1020 to 1080, and the number of features varies from 7 to 25, including the class variable.

For each dataset we evaluated some relevant two-component mixtures pairs B_0–B_1, namely, tree–tree, tree–C2G, C2G–tree and C2G–C2G. These mixtures were tested with the mfCLL scoring criterion, with and without the MDL penalty. In order to provide a baseline for comparison of the proposed hybrid models, we also evaluated them with (generative) LL and MDL scoring criteria, as firstly proposed by Barash *et al.* [1]. These resulted in the sixteen candidate models, presented in Table 1, eight of them hybrid and the other eight generatively learned. We improved the performance of BNs by smoothing the parameter estimates according to a Dirichlet prior [23], which in practice corresponds to adding a certain amount of pseudo-counts uniformly. The smoothing parameter was set to 0.5 as it is common practice.

The accuracy of each mixture is defined as the percentage of successful predictions on the test sets of each dataset. Accuracy was measured via stratified five-fold cross-validation, using the methods described by Kohavi [27]. Throughout the experiments, we used the same cross-validation folds for every classifier. Scatter plots of the accuracies of the proposed methods against the others are depicted in Figure 1. Points above the diagonal line represent cases where the method shown in the vertical axis performs better than the one on the horizontal axis. For space considerations, only eight (one against seven) out of the sixteen models tested are depicted and compared with the best mixture, C2G–C2G–mfCLL$^{\mathrm{MDL}}$. Nonetheless, the results for the remaining models are discussed along this section.

As suggested in [14], we also compared the performance of the classifiers using Wilcoxon signed-rank tests. This test is applicable when paired classification accuracy differences, along the datasets, are independent and non-normally distributed. Alternatively, a paired t-test could have been used but we applied the former as the Wilcoxon signed-rank test is more conservative than the paired t-test. We concluded that hybrid mixtures of C2G models significantly outperformed the remaining mixtures. However, as expected, the MDL penalty was needed in order to control the complexity of the resulting C2G local models. Results comparing the mixtures analyzed in Figure 1 with the four most

		Backgorund		Foreground	
Learning	Abbreviation	Model	Score	Model	Score
hybrid	tree–tree–mfCLL	tree	LL	tree	mfCLL
	tree–C2G–mfCLL	tree		C2G	
	C2G–tree–mfCLL	C2G		tree	
	C2G–C2G–mfCLL	C2G		C2G	
	tree–tree–mfCLL$^{\mathrm{MDL}}$	tree	MDL	tree	mfCLL$^{\mathrm{MDL}}$
	tree–C2G–mfCLL$^{\mathrm{MDL}}$	tree		C2G	
	C2G–tree–mfCLL$^{\mathrm{MDL}}$	C2G		tree	
	C2G–C2G–mfCLL$^{\mathrm{MDL}}$	C2G		C2G	
generative	tree–tree–LL	tree	LL	tree	LL
	tree–C2G–LL	tree		C2G	
	C2G–tree–LL	C2G		tree	
	C2G–C2G–LL	C2G		C2G	
	tree–tree–MDL	tree	MDL	tree	MDL
	tree–C2G–MDL	tree		C2G	
	C2G–tree–MDL	C2G		tree	
	C2G–C2G–MDL	C2G		C2G	

Table 1
A total of sixteen multinets were used in the experiments. In the abbreviated name the first word is the model employed in the background learning whereas the second concerns the foreground. The third word indicates the score used for learning the foreground model. The background model uses the same score as the foreground if the model is generative, and the corresponding generative version when the model is hybrid.

promising hybrid mixtures are depicted in Table 2. Each entry in the table has the p-value of the significance test for the corresponding pairs of classifiers. The double arrow points to the best classifier, in terms of classification rate, with p-value smaller than 0.05.

Multinet **Score**	tree–C2G mfCLL$^{\mathrm{MDL}}$	C2G–C2G mfCLL	tree–C2G mfCLL	C2G–tree mfCLL$^{\mathrm{MDL}}$	C2G–tree mfCLL	tree-tree MDL	tree-tree LL
C2G–C2G mfCLL$^{\mathrm{MDL}}$	2.6×10^{-2} $\Leftarrow$	1.7×10^{-16} $\Leftarrow$	1.5×10^{-7} $\Leftarrow$	3.1×10^{-2} $\Leftarrow$	7.2×10^{-10} $\Leftarrow$	7.2×10^{-10} $\Leftarrow$	6.3×10^{-15} $\Leftarrow$
tree–C2G mfCLL$^{\mathrm{MDL}}$		5.7×10^{-16} $\Leftarrow$	8.2×10^{-6} $\Leftarrow$	1.9×10^{-2} $\Leftarrow$	5.0×10^{-9} $\Leftarrow$	5.0×10^{-9} $\Leftarrow$	2.2×10^{-14} $\Leftarrow$
C2G–C2G mfCLL			6.8×10^{-16} $\Uparrow$	6.8×10^{-16} $\Uparrow$	1.3×10^{-14} $\Uparrow$	1.3×10^{-14} $\Uparrow$	6.7×10^{-13} $\Uparrow$
tree–C2G mfCLL				2.7×10^{-6} $\Uparrow$	4.8×10^{-7} $\Leftarrow$	1.4×10^{-2} $\Leftarrow$	1.4×10^{-2} $\Leftarrow$

Table 2. Wilcoxon signed-rank tests for the results obtained by multinets over 89 datasets with biologically validated TFBSs. Each entry of the table has the *p*-value of the significance test for the corresponding pair of classifiers. The double arrow points to the superior learning algorithm, in terms of classification rate, with a *p*-value smaller than 0.05. It is clear that the C2G–C2G–mfCLL$^{\mathrm{MDL}}$ outperformed all other multinets with high statistical significance.

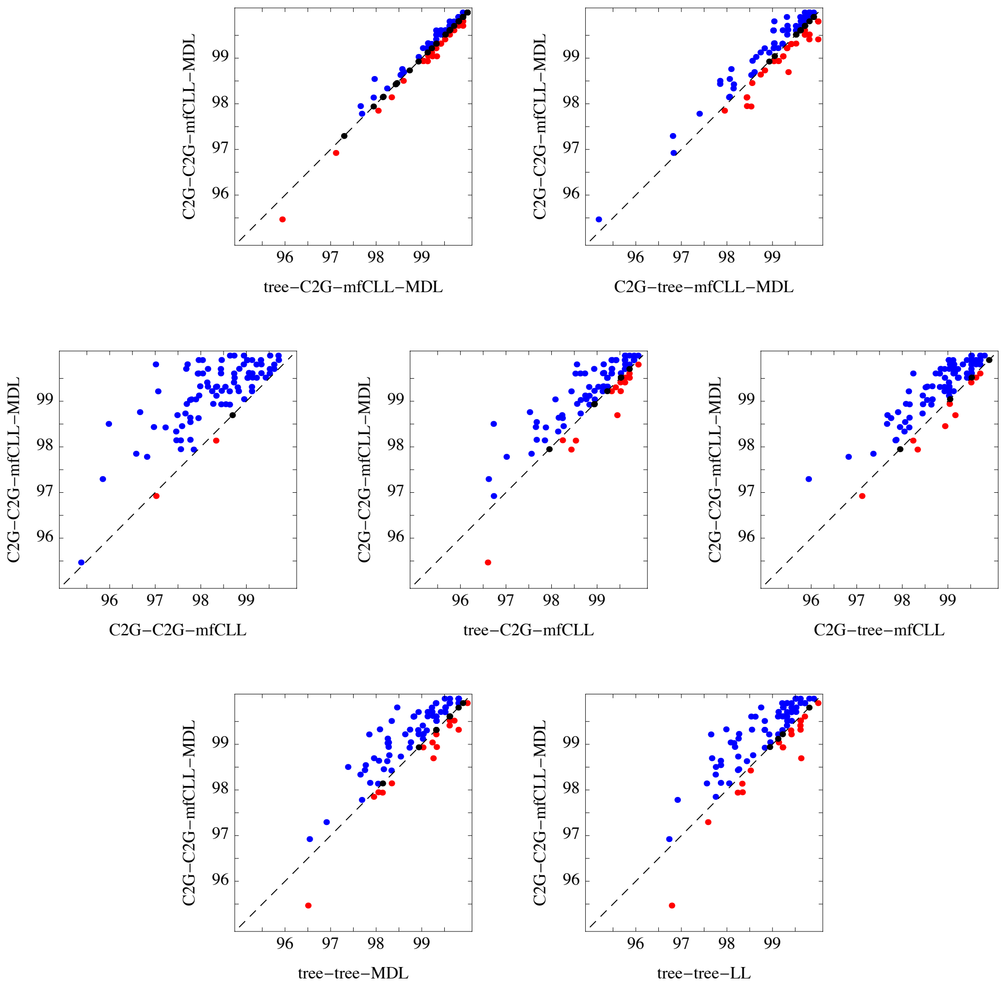

Figure 1. Scatter plots of the accuracy of different multinet classifiers compared with the penalized hybrid mixture C2G–C2G–mfCLL$^{\mathrm{MDL}}$ over 89 datasets with biologically validated TFBSs. The first two plots correspond to other penalized hybrid mixtures, the following three correspond to non-penalized hybrid mixtures, and the last two to generative mixtures. Points above the diagonal line represent the cases where the method shown on the vertical axis performs better than the one on the horizontal axis. C2G–C2G–mfCLL$^{\mathrm{MDL}}$ was the classifier with the best performance.

From Table 2 it is clear that C2G−C2G−mfCLL is overfitting whereas the same is not true, at least at the same scale, with tree–C2G–mfCLL. This points out that overfitting is mainly occurring in the background model B_0. However, despite the fact that tree–C2G–mfCLL performed better than C2G–

C2G–mfCLL, Table 2 also shows that higher accuracies are achieved with the same two-component mixture models but with the penalized versions of mfCLL. Actually, the combination of mfCLL$^{\mathrm{MDL}}$ scoring criterion with two-component mixtures of C2G models (first line of Table 2) performs better than all the other considered classifiers. We conclude that hybrid 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.

Although not depicted in Figure 1 nor presented in Table 2, we also directly compared tree–tree–mfCLL with tree–tree–LL classifiers, and tree–tree–mfCLL$^{\mathrm{MDL}}$ with tree–tree–MDL, in order to understand the benefits of using the mfCLL score without the degree of freedom introduced by the C2G model. In this way, we performed a comparison with previous results on using BN mixtures to model TFBSs [1]. Results showed that tree–tree–mfCLL significantly outperformed tree–tree–LL with a p-value of 2.08×10^{-6} and that tree–tree–mfCLL$^{\mathrm{MDL}}$ also performed significantly better than tree–tree–MDL with a p-value of 1.37×10^{-6}. Furthermore, the hybrid mixture that provided the best results, C2G–C2G–mfCLL$^{\mathrm{MDL}}$, outperformed with statistical significance both tree–tree–mfCLL, tree–tree–mfCLL$^{\mathrm{MDL}}$, and mixtures of PSSMs. We conclude then that a discriminative scoring criterion such as mfCLL, with or without MDL penalty, is advantageous in classification tasks when compared to their generative counterparts (LL and MDL).

Our results support that hybrid multinets are suitable for TFBS discrimination, and confirm those of Barash *et al.* [1] that had already noticed that generative multinets outperform plain BN models when modeling TFBSs.

4.1.2 Evaluating other state-of-the-art models

The usage of BNs to model TFBS is linked to the fact that they provide an intuitive and humane readable model. Nevertheless, discriminating TFBSs from background is mainly a classification task and so any classifier can be used, independently of providing or not a TFBS model capable of being interpreted or biologically validated. Therefore, we also compared hybrid CκG-mixtures with state-of-the-art classifiers, namely, *support vector machines* (SVM), *logistic regression* (LogR), *decision trees* (DT) and *k-nearest neighbor* (k-NN).

All aforementioned state-of-the-art classifiers were tested in WEKA java package [22] and are summarized in Table 3. Concerning SVM models, we used three different kernels: (i) a linear kernel (`SMO` implementation in WEKA), which we henceforward denote by SVM; (ii) a polynomial kernel (`SMO` implementation in WEKA with `PolyKernel` and exponent parameter `E=2`), denoted by SVM2; and (iii) a *radial basis function* (RBF) kernel also known as Gaussian kernel (`SMO` implementation in WEKA with `RBFKernel`), which we denote by SVMG. Following the canon in the literature [25], we used a grid-search to find the optimal penalty parameter C and the optimal RBF kernel parameter γ, using cross-validation. More specifically, for linear, polynomial, and RBF kernels, 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 additionally selected γ from $[10^{-3}, 10^{-2}, 10^{-1}, 1, 10]$ in a similar manner.

In what concerns logistic regression and decision trees, `Logistic` and `J48` implementations from WEKA were used, respectively, with default parameters. Concerning k-NN classifier, we used $k = 3$ (`IBk` implementation in WEKA with parameter $K = 3$) and $k = 5$ (`IBk` implementation in WEKA with parameter $K = 5$). Finally, for TAN-fCLL, we improved its performance using *Dirichlet priors* (see [23]) to smooth the network parameters. We achieve this purpose by setting the `alpha` parameter to 0.5. In practice, this is the default value for this parameter, and the value for which we obtained the highest average accuracy among all classifiers. The rest of the experimental procedure was similar to the one described in the previous section. Results are illustrated in Figure 2 and Table 4.

Abbreviation	**Classifier**	**Implementation**
DT	Decision tree	`J48` implementation from WEKA
3-NN	3-Nearest neighbor	`IBk` (K=3) implementation from WEKA
5-NN	5-Nearest neighbor	`IBk` (K=5) implementation from WEKA
SVM	Support vector machine with linear lernel	`SMO` implementation from WEKA
SVM2	Support vector machine with polynimial kernel	`SMO` with `PolyKernel` (E=2) implementation from WEKA
SVMG	Support vector machine with Guassin kernel	`SMO` with `RBFKernel` implementation from WEKA
LogR	Logistic regression	`Logistic` implementation from WEKA
TAN-fCLL	Tree augmented naive BN classifier with fCLL scoring criterion	`TAN` implementation from [8] (score=fCLL)

Table 3
Discriminative state-of-the-art classifiers used in the experiments. Parameters, if different from their defaults, are given parenthetically in the last column of the table. Concerning SVMs, a grid-search was used to find the optimal penalty parameter C, and the optimal RBF kernel parameter γ using cross-validation. More specifically, for linear, polynomial, and RBF kernels, C was selected from $[10^{-1}, 1, 10, 10^2]$ by using 5-fold cross validation on the training set. For the RBF kernel γ was also selected from $[10^{-3}, 10^{-2}, 10^{-1}, 1, 10]$ in a similar manner.

From the analysis of Figure 2 and Table 4 it is clear that C2G–C2G–mfCLL$^{\mathrm{MDL}}$ and SVM2 significantly outperformed all other state-of-the-art classifiers. C2G–C2G–mfCLL$^{\mathrm{MDL}}$ also performed better than SVM2 although the difference was not statistical significant. The main advantage of C2G–C2G–mfCLL$^{\mathrm{MDL}}$ over SVM2 is its computational cost, as SVM2 needs to tune its parameters with a time-consuming grid-search procedure, taking 2 to 3 orders of magnitude more time than C2G–C2G–mfCLL$^{\mathrm{MDL}}$.

The other classifiers behaved somewhat as expected. Gaussian SVMs significantly outperformed all other classifiers, with the exception of mfCLL$^{\mathrm{MDL}}$ and

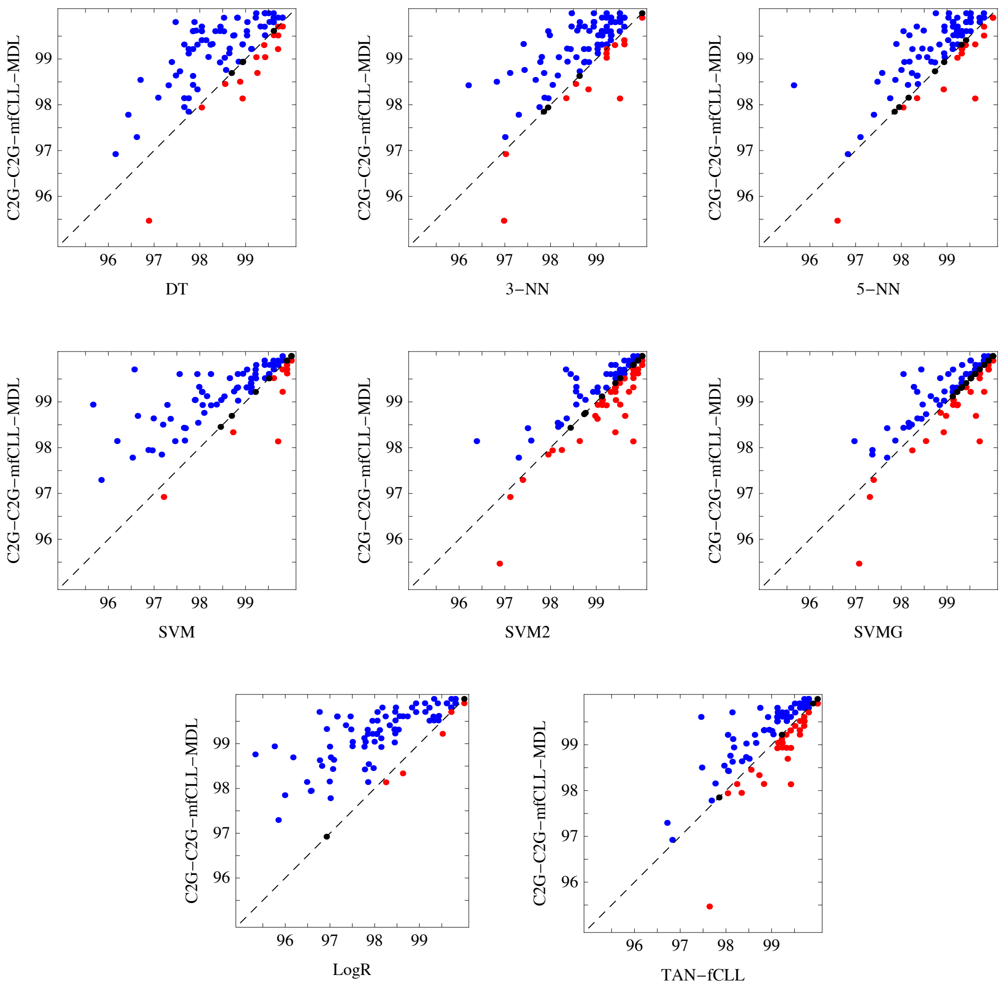

Figure 2. Scatter plots of the accuracy of different state-of-the-art classifiers compared with C2G–C2G–mfCLL$^{\mathrm{MDL}}$, that was the mixture that performed better than all other mixtures (in Table 1) over the 89 datasets with biologically validated TFBSs (see Table 2). Points above the diagonal line represent the cases where the method shown on the vertical axis performs better than the one on the horizontal axis. The C2G–C2G–mfCLL$^{\mathrm{MDL}}$ was the best classifier.

polynomial SVMs. Among linear SVMs, logistic regression, k-nearest neighbors and decision trees, the 5-NN was the one that performed the best, followed by linear SVM, 3-NN and DT. Surprisingly, LogR attained the worst results in this setup.

Classifier	DT	3-NN	5-NN	SVM	SVM2	SVMG	LogR	TAN–fCLL
C2G–C2G–mfCLL$^{\mathrm{MDL}}$	1.8×10^{-11} $\Leftarrow$	2.1×10^{-11} $\Leftarrow$	9.3×10^{-10} $\Leftarrow$	1.2×10^{-11} $\Leftarrow$	0.25 $\leftarrow$	3.5×10^{-3} $\Leftarrow$	1.2×10^{-15} $\Leftarrow$	1.0×10^{-4} $\Leftarrow$
DT		1.9×10^{-2} $\Uparrow$	6.4×10^{-4} $\Uparrow$	0.18 $\uparrow$	1.3×10^{-11} $\Uparrow$	1.5×10^{-4} $\Uparrow$	6.4×10^{-5} $\Leftarrow$	2.0×10^{-8} $\Uparrow$
3-NN			1.5×10^{-4} $\Uparrow$	0.25 $\leftarrow$	1.7×10^{-11} $\Uparrow$	1.6×10^{-8} $\Uparrow$	4.7×10^{-8} $\Leftarrow$	3.5×10^{-6} $\Uparrow$
5-NN				1.7×10^{-2} $\Leftarrow$	1.9×10^{-9} $\Uparrow$	2.1×10^{-6} $\Uparrow$	4.4×10^{-10} $\Leftarrow$	1.3×10^{-3} $\Uparrow$
SVM					1.9×10^{-11} $\Uparrow$	2.7×10^{-11} $\Uparrow$	9.6×10^{-13} $\Leftarrow$	5.7×10^{-4} $\Uparrow$
SVM2						6.8×10^{-3} $\Leftarrow$	8.0×10^{-15} $\Leftarrow$	2.3×10^{-4} $\Leftarrow$
SVMG							1.9×10^{-15} $\Leftarrow$	9.4×10^{-3} $\Leftarrow$
LogR								1.1×10^{-14} $\Uparrow$

Table 4. Wilcoxon signed-rank tests for the results obtained by state-of-the-art classifiers over 89 datasets with biologically validated TFBSs. Each entry of the table has the p-value of the significance test for the corresponding pair of classifiers. The arrow points to the superior learning algorithm, in terms of classification rate. A double arrow is used when the difference is significant with p-value smaller than 0.05. It is clear that the C2G–C2G–mfCLL$^{\mathrm{MDL}}$ and SVM2 outperformed all other classifiers with high statistical significance. Given that C2G–C2G–mfCLL$^{\mathrm{MDL}}$ requires no parameter optimization and low learning times, it seems to be the best strategy.

4.2 Clinical data

We also tested the proposed hybrid classifier C2G–C2G–mfCLL$^{\mathrm{MDL}}$ over eight datasets from the UCI repository [30]. We chose eight diagnosis datasets that were within the scope of application of the proposed method—a background characterizing the values of the attributes for healthy individuals, and a foreground with the attributes for patients with a particular disease. These datasets are described in Table 5.

	Dataset	**Features**	**Train**	**Test**
1	breast	10	683	CV-5
2	cleve	14	296	CV-5
3	diabetes	9	768	CV-5
4	echo	12	132	CV-5
5	hepatitis	20	80	CV-5
6	mammo	6	961	CV-5
7	pima	9	768	CV-5
8	thyroid	5	215	CV-5

Table 5
Description of UCI datasets used in the experiments. All of these eight datasets were collected from clinical data: 1) breast cancer; 2) heart disease from Cleveland patiens; 3) diabetes patient records retrieved from automatic electronic recording device and paper records; 4) echocardiogram from patients that suffered heart attacks; 5) hepatitis disease; 6) mammographic masses from breast cancer screening; 7) Pima Indians diabetes; and 8) thyroid disease.

The continuous-valued attributes in the datasets were discretized in a supervised manner using the entropy-based method proposed in [17]. For this task we used the WEKA package. Supervised discretization was performed via `weka.filters.supervised.attribute.Discretize`, with default parameters. Moreover, instances with missing values were removed from the datasets.

The accuracy of each classifier was measured via stratified 5-fold cross-validation. Throughout the experiments, we used exactly the same folds, hence, the same information was available for training and testing all classifiers. The accuracy results are presented in Table 6. Similarly to the previous section, we compared the performance of the classifiers using Wilcoxon signed-rank tests (Table 7).

Dataset	C2G–C2G mfCLL$^{\text{MDL}}$	DT	3-NN	5-NN	SVM	SVM2	SVMG	LogR	TAN-fCLL
breast	97.66	95.90	96.93	96.93	97.51	96.05	96.63	96.63	97.66
cleve	82.77	76.69	80.41	82.77	82.09	72.97	78.38	81.42	82.77
diabetes	79.17	77.60	77.86	77.73	77.47	76.56	77.86	78.65	78.91
echo	79.03	79.03	74.19	79.03	75.81	79.03	79.03	70.97	72.58
hepatitis	93.75	85.00	91.25	92.50	83.75	87.50	87.50	78.75	90.00
mammo	83.86	83.37	78.31	80.36	81.69	85.06	84.46	82.41	82.89
pima	78.39	77.21	76.82	76.69	78.91	76.95	77.08	78.26	78.52
thyroid	94.88	94.42	93.49	92.56	94.88	94.88	94.88	88.84	94.42

Table 6
Accuracy attained by each classifier over the eight datasets described in Table 5. The best classifier was the C2G–C2G–mfCLL$^{\text{MDL}}$ followed by TAN-fCLL.

From the analysis of tables 6 and 7 it is clear that the hybrid mixture C2G–C2G–mfCLL$^{\text{MDL}}$ outperformed significantly all the other classifiers. The TAN-fCLL classifier also performed well however, it only attained statistical significant outperformance against LogR.

Classifier	DT	3-NN	5-NN	SVM	SVM2	SVMG	LogR	TAN-fCLL
C2G–C2G–mfCLL$^{\mathrm{MDL}}$	0.011	0.007	0.018	0.021	0.029	0.029	0.007	0.030
	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$
DT		0.500	0.336	0.417	0.417	0.015	0.264	0.176
		$\equiv$	$\uparrow$	$\uparrow$	$\uparrow$	$\Uparrow$	$\leftarrow$	$\uparrow$
3-NN			0.102	0.117	0.500	0.336	0.363	0.117
			$\uparrow$	$\uparrow$	$\equiv$	$\uparrow$	$\leftarrow$	$\uparrow$
5-NN				0.472	0.312	0.500	0.181	0.400
				$\leftarrow$	$\leftarrow$	$\equiv$	$\leftarrow$	$\uparrow$
SVM					0.466	0.336	0.092	0.221
					$\uparrow$	$\uparrow$	$\leftarrow$	$\uparrow$
SVM2						0.140	0.264	0.221
						$\uparrow$	$\leftarrow$	$\uparrow$
SVMG							0.136	0.312
							$\leftarrow$	$\uparrow$
LogR								0.007
								$\Uparrow$

Table 7

Wilcoxon signed-rank tests for the results obtained by state-of-the-art classifiers over the datasets in Table 5. Each entry of the table has the p-value of the significance test for the corresponding pair of classifiers. The arrow points to the best of the two learning algorithms, in terms of classification rate. A double arrow is used when the difference is significant with p-value smaller than 0.05. The symbol $\equiv$ means that the models attained equivalent results. The C2G–C2G–mfCLL$^{\mathrm{MDL}}$ was the best strategy.

5 Conclusions

In this paper we proposed a new scoring criterion, which we called mfCLL, for learning two-component mixtures of BNs. The new score is used to discriminatively learn the concepts in one of the components of the mixture—the foreground. On the other hand, background concepts are learned generatively, providing an overall hybrid procedure for binary classification tasks. This new score is decomposable and the parameters that maximize it are known, allowing for score-based learning procedures to be employed very efficiently.

The benefits of this new scoring criterion were evaluated on a large suite of 89 benchmark datasets from computational biology for the task of TFBS representation. Results showed that among all candidate multinets, the hybrid C2G–C2G–mfCLL$^{\mathrm{MDL}}$ mixture, was the one that provided a superior discriminant power to distinguish TFBSs from non-binding regions in comparison with the existing generative models. In addition, a comparison with other state-of-the-art classifiers, showed that C2G–C2G–mfCLL$^{\mathrm{MDL}}$ significantly outperforms SVMs with linear and Guassian kernels, logistic regression, decision trees, and k-nearest neighbors for $k = 3$ and $k = 5$. Moreover, C2G–C2G–mfCLL$^{\mathrm{MDL}}$ showed to behave similarly to polynomial SVMs. Notwithstanding, learning mfCLL-based mixtures is considerably more time-efficient than learning SVMs, taking 2 to 3 orders of magnitude less time for the 89 considered datasets.

In addition, we also gauged the merits of C2G–C2G–mfCLL$^{\mathrm{MDL}}$ with eight UCI diagnosis datasets within the scope of application of the proposed method. The results showed that C2G–C2G–mfCLL$^{\mathrm{MDL}}$ outperformed significantly all other classifiers.

We conclude that the hybrid C2G–C2G–mfCLL$^{\mathrm{MDL}}$ classifier constitutes a good choice when the data is split in foreground and background, where the background is noisy and consists of a possible miscellaneous of concepts whereas the foreground contains a single and well defined concept.

Directions for future work include the study of the asymptotic behavior of mfCLL and the usage of mfCLL in unsupervised learning.

Acknowledgements and funding

The authors are grateful to the invaluable comments by the anonymous referees to a previous version of the paper. This work was partially supported by Fundação para a Ciência e Tecnologia (FCT), under grant PEst-OE/EEI/LA0008/2013, and by the projects NEUROCLINOMICS (PTDC/EIA-EIA/111239/2009), InteleGen (PTDC/DTP-FTO/1747/2012), and ComFormCrypt (PTDC/EIA-CCO/113033/2009), also funded by FCT.

References

[1] Y. Barash, G. Elidan, N. Friedman, and T. Kaplan. Modeling dependencies in protein-DNA binding sites. In *Proc. RECOMB'03*, pages 28–37, 2003.

[2] J. Bilmes. Dynamic Bayesian multinets. In *Proc. UAI'00*, pages 38–45, 2000.

[3] Christopher M. Bishop and Julia Lasserre. Generative or Discriminative? Getting the Best of Both Worlds. *BAYESIAN STATISTICS*, 8:3–24, 2007.

[4] G. Bouchard and B. Triggs. The trade-off between generative and discriminative classifiers. In *Proc. COMPSTAT'04*, pages 721–728. Physica-Verlag, 2004.

[5] A. M. Carvalho. Scoring functions for learning Bayesian networks. Technical report, INESC-ID Tec. Rep. 54/2009, 2009.

[6] A. M. Carvalho and A. L. Oliveira. Learning Bayesian networks consistent with the optimal branching. In *Proc. ICMLA'07*, pages 369–374, 2007.

[7] 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.

[8] 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(Jul):2181–2210, 2011.

[9] R. Castelo and A. Siebes. Priors on network structures. biasing the search for bayesian networks. *Int. J. Approx. Reasoning*, 24(1):39–57, 2000.

[10] D. M. Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

[11] 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.

[12] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[13] 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.

[14] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[15] 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.

[16] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240, 1967.

[17] 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.

[18] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

[19] D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, 82(1-2):45–74, 1996.

[20] 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.

[21] D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *Proc. ICML'04*, pages 46–53, 2004.

[22] 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.

[23] 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.

[24] Alex Holub and Pietro Perona. A discriminative framework for modelling object classes. In *Proc. IEEE CVPR'05*, pages 664–671. IEEE Computer Society, 2005.

[25] 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.

[26] M. Hutter. Discrete MDL predicts in total variation. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *NIPS 2009*, pages 817–825. Curran Associates, Inc., 2009.

[27] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. IJCAI'95*, pages 1137–1145, 1995.

[28] D. Koller and N. Friedman. *Probabilistic Graphical models: Principles and techniques.* MIT Press, 2009.

[29] M. Meila and M. I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.

[30] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998.

[31] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Proc. NIPS'01*, pages 841–848, 2001.

[32] R. A. O'Flanagan, G. Paillard, R. Lavery, and A. M. Sengupta. Non-additivity in protein-DNA binding. *Bioinformatics*, 21(10):2254–2263, 2005.

[33] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[34] S. V. Pemmaraju and S. S. Skiena. *Computational discrete mathematics: Combinatorics and graph theory with Mathematica.* Cambridge University Press, 2003.

[35] 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.

[36] Rajat Raina, Yirong Shen, Andrew Y. Ng, and Andrew McCallum. Classification with hybrid generative/discriminative models. In *NIPS*, 2003.

[37] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431, 1983.

[38] J. Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society, Series B*, 49(3):223–239, 1987.

[39] 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.

[40] J. Su and H. Zhang. Full Bayesian network classifiers. In *Proc. ICML'06*, pages 897–904, 2006.

[41] J. Su, H. Zhang, C. X. Ling, and S. Matwin. Discriminative parameter learning for Bayesian networks. In *Proc ICML'08*, pages 1016–1023, 2008.

[42] T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proc. UAI'90*, pages 255–270, 1990.

[43] E. Wingender, X. Chen, E. Fricke, R. Geffers, R. Hehl, I. Liebich, M. Krull, V. Matys, H. Michael, R. Ohnhuser, M. Prss, F. Schacherer, S. Thiele, and S. Urbach. The TRANSFAC system on gene expression regulation. *Nucleic Acids Research*, 29(1):281–283, 2001.

[44] 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.

***Author Biography**

Alexandra M. Carvalho received the PhD degree in Computer Science in 2011. She is an Assistant professor of IST and a researcher in the Pattern and Image Analysis Group of the Instituto de Telecomunicações. Her research interests include pattern recognition, machine learning, combinatorics and algorithms.

Pedro Adão received his PhD degree in Mathematics (2006). He is currently an Assistant Professor in the Computer Science Department, IST, and a researcher of the Security and Quantum Information Group of the Instituto de Telecomunicações. His research interests include security and quantum cryptography.

Paulo Mateus, PhD (2001) and Habilitation (2005), is an Associate Professor from the Mathematics Department of Instituto Superior Técnico, Technical University of Lisbon, Portugal. He is a researcher from Instituto de Telecomunicações where he coordinates the Security and Quantum Information Group. He is also member of the Managing Board of the European Network and Information Security Agency and vice-president of Centro Internacional de Matemática. He was awarded the IBM scientific prize, Portugal, in 2005. His research interests include quantum information theory and discrete structures. He has been author and co-author of around 60 international publications.

LaTeX Source Files

Click here to download LaTeX Source Files: sources-vs2.zip