Noname manuscript No.
(will be inserted by the editor)

Efficient Parameter Learning of Bayesian Network Classifiers

Nayyar A. Zaidi · Geoffrey I. Webb · Mark J. Carman · François Petitjean

Received: date / Accepted: date

Abstract It has recently been shown that it can be very productive to learn models that combine both generative and discriminative parameter estimation. On the one hand, generative approaches address the estimation of the joint distribution – $\mathrm{P}(y, \mathbf{x})$, and are very *efficient* and on the other hand, discriminative approaches address the estimation of the posterior distribution – and, are more *effective* for classification, since they model $\mathrm{P}(y|\mathbf{x})$ directly. However, discriminative approaches are less computationally efficient as the normalization factor in the conditional log-likelihood precludes the derivation of closed-form estimation of parameters. In this paper, we take the widely used model of the joint distribution – Bayesian networks – and show how parameter learning can be done discriminatively but efficiently for classification. The contributions of this paper are twofold – first, we propose a unified theoretical framework to characterize the parameter learning task for Bayesian network classifiers and second, we introduce a combined generative/discriminative parameter learning method for Bayesian network classifiers. We conduct an extensive set of experiments on 72 standard datasets and demonstrate that our proposed parameterization provides an efficient discriminative parameter learning scheme that outperforms other state-of-the-art parameterizations.

1 Introduction

The efficient training of *Bayesian Network Classifiers* has been the topic of much recent research [1,3,6,10,13,16,22,24]. Two paradigms predominate [11]. One can optimize the log-likelihood (LL). This is traditionally called *generative learning*. The goal is to obtain parameters characterizing the joint distribution in the form of local conditional distributions and obtain the class-conditional probabilities by using the Bayes rule. Alternatively, one can optimize the

Faculty of Information Technology, Monash University, VIC 3800, Australia
E-mail: {nayyar.zaidi, geoff.webb, mark.carman, francois.petitjean}@monash.edu

conditional-log-likelihood (CLL) – known as *discriminative learning*. The goal is to directly estimate the parameters associated with the class-conditional distribution – $\mathrm{P}(y|\mathbf{x})$.

Naive Bayes (NB) is a Bayesian network BN that specifies independence between attributes given the class. Recent work has shown that placing a per-attribute-value-per-class-value weight on probabilities in NB (and learning these weights by optimizing the CLL) leads to an alternative parameterization of vanilla Logistic Regression (LR) [23]. Introduction of these weights also makes it possible to relax NB's conditional independence assumption and thus to create a classifier with lower bias [14,23]. In this paper, we generalize this idea to the general class of BN classifiers. Like NB, any given BN structure encodes assumptions about conditional independencies between the attributes and will result in error if they do not hold in the data. Optimizing the log-likelihood in this case will result in suboptimal performance for classification [6,9,21] and one should either optimize directly the CLL by learning the parameters of the class-conditional distribution or by placing weights on the probabilities and learn these weights by optimizing the CLL.

We start by introducing a unified theoretical framework for the learning of the parameters of Bayesian network classifiers. Building on previous work by [6,8,17,19,24], this framework allows us to lay out the different techniques in a systematic manner; highlighting similarities, distinctions and equivalences. Then we introduce a new parameterization – *weighted Bayesian Network Classifiers* – that combines the efficiency of the generative approach by pre-conditioning the weights, and the effectiveness of the discriminative approach by optimizing the CLL. It is based on a two-step learning process:

1. Generative step: We minimize the LL to obtain parameters for all local conditional distributions in the BN.
2. Discriminative step: We associate a weight with each parameter learned in the generative step and re-parameterize the class-conditional distribution in terms of these weights (and of the fixed generative parameters). We can then discriminatively learn these weights by optimizing the CLL.

It is interesting to note that, at first sight, one could question the necessity of the generative step, because we know that with or without preconditioning, (when optimizing a convex objective function such as CLL) the parameters have to converge to the same point in the search space – preconditioning has the effect of re-scaling the axis. This is a valid question, to which this paper gives a direct answer: we show that our two-step formalization of the parameter learning task for BN is actually a re-parameterization of the one step (discriminative) learning problem but with faster convergence of the discriminative optimization procedure. In the experimental section, we complement our theoretical framework with an empirical analysis over 72 domains; the results demonstrate the superiority of our approach over the state of the art.

The rest of this paper is organized as follows. In Section 2, we present our proposed unified framework for parameter learning of Bayesian network classifiers. We also give the formulation for class-conditional Bayesian Network

models (CCBN) in this section. Two well-used parameterizations of class-conditional Bayesian networks are given in Sections 3 and 4, respectively. In Section 5, we present our proposed parameterization of CCBN. In Section 6, we discuss some related work to this research. Experimental analysis is conducted in Section 7. We conclude in Section 8 with some pointers to future work.

2 A Unified Theoretical Framework for Parameter Learning of Bayesian Network Classifiers

We start by discussing Bayesian Network classifiers in the following section.

2.1 Bayesian Network Classifiers

A BN $\mathcal{B} = \langle \mathcal{G}, \Theta \rangle$, is characterized by the structure $\mathcal{G}$ (a directed acyclic graph, where each vertex is an attribute, X), and a set of parameters Θ, that quantifies the dependencies within the structure. The parameter Θ, contains a set of parameters for each vertex in $\mathcal{G}$: $\theta_{x_i|\Pi_i(\mathbf{x})}$, where $\Pi_i(.)$ is a function which given the datum $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ as its input, returns the values of the attributes which are the parents of node i in structure $\mathcal{G}$. For notational simplicity, instead of writing $\theta_{X_i=x_i|\Pi_i(\mathbf{x})}$, we write $\theta_{x_i|\Pi_i(\mathbf{x})}$. A BN $\mathcal{B}$ computes the joint probability distribution as: $\mathrm{P}_{\mathcal{B}}(\mathbf{x}) = \prod_{i=0}^{n} \theta_{x_i|\Pi_i(\mathbf{x})}$. The goal of developing BN is to predict the value of some class variable, say X_0. We will assume that the first attribute is the class attribute and denote it with Y (i.e., $X_0 = Y$), and denote a value for that attribute by y, where $y \in \mathcal{Y}$. For a BN defining $\mathrm{P}_{\mathcal{B}}(\mathbf{x}, y)$, the corresponding conditional distribution $\mathrm{P}_{\mathcal{B}}(y|\mathbf{x})$ can be written as:

$$\mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = \frac{\mathrm{P}_{\mathcal{B}}(y, \mathbf{x})}{\mathrm{P}_{\mathcal{B}}(\mathbf{x})} = \frac{\theta_{y|\Pi_0(\mathbf{x})} \prod_{i=1}^{n} \theta_{x_i|y,\Pi_i(\mathbf{x})}}{\sum_{y'}^{|\mathcal{Y}|} \theta_{y'|\Pi_0(\mathbf{x})} \prod_{i=1}^{n} \theta_{x_i|y',\Pi_i(\mathbf{x})}}. \tag{1}$$

If the class attribute does not have any parents, we write: $\theta_{y|\Pi_0(\mathbf{x})} = \theta_y$.

Given a set of data points $\mathcal{D} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$, the Log-Likelihood (LL) of $\mathcal{B}$ is:

$$\begin{aligned} \mathrm{LL}(\mathcal{B}) &= \sum_{j=1}^{N} \log \mathrm{P}_{\mathcal{B}}(y^{(j)}, \mathbf{x}^{(j)}), \\ &= \sum_{j=1}^{N} \left(\log \theta_{y^{(j)}|\Pi_0(\mathbf{x}^{(j)})} + \sum_{i=1}^{n} \log \theta_{x_i^{(j)}|\Pi_i(\mathbf{x}^{(j)})} \right), \end{aligned} \tag{2}$$

$$\text{with} \quad \sum_{y \in \mathcal{Y}} \theta_{y|\Pi_0(\mathbf{x})} = 1, \quad \text{and} \quad \sum_{x_i \in \mathrm{dom}(X_i)} \theta_{x_i|\Pi_i(\mathbf{x})} = 1. \tag{3}$$

Maximizing Equation 2 to optimize the parameters (θ) is known as maximum-likelihood estimation.

Theorem 1 *With constraints in Equation 3, Equation 2 is maximized when* $\theta_{x_i|\Pi_i(\mathbf{x})}$ *corresponds to empirical estimates of probabilities from the data, that is,* $\theta_{y|\Pi_0(\mathbf{x})} = \mathrm{P}_{\mathcal{D}}(y|\Pi_0(\mathbf{x}))$ *and* $\theta_{x_i|\Pi_i(\mathbf{x})} = \mathrm{P}_{\mathcal{D}}(x_i|\Pi_i(\mathbf{x}))$.

Proof See Appendix A.

The parameters obtained by maximizing Equation 2 (and fulfilling the constraints in Equation 3) are typically known as 'Generative' estimates of the probabilities.

2.2 Class-Conditional BN (CCBN) Models

Instead of maximizing Equation 2, for classification problems, maximizing Conditional Log-Likelihood (CLL) is generally a more effective objective function since it directly optimizes the mapping from features to class labels. The CLL can be defined as:

$$\mathrm{CLL}(\mathcal{B}) = \sum_{j=1}^{N} \log \mathrm{P}_{\mathcal{B}}(y^{(j)}|\mathbf{x}^{(j)}),$$

which is equal to:

$$\begin{aligned}
&= \sum_{j=1}^{N} \left(\log \mathrm{P}_{\mathcal{B}}(y^{(j)}, \mathbf{x}^{(j)}) - \log \sum_{y'}^{|\mathcal{Y}|} \mathrm{P}_{\mathcal{B}}(y', \mathbf{x}^{(j)}) \right) \\
&= \sum_{j=1}^{N} \left(\log \theta_{y^{(j)}|\Pi_0(\mathbf{x}^{(j)})} + \sum_{i=1}^{n} \log \theta_{x_i^{(j)}|\Pi_i(\mathbf{x}^{(j)})} \right) - \\
&\qquad \log \left(\sum_{y'}^{|\mathcal{Y}|} \theta_{y'|\Pi_0(\mathbf{x}^{(j)})} \prod_{i=1}^{n} \theta_{x_i|y',\Pi_i(\mathbf{x}^{(j)})} \right). \qquad (4)
\end{aligned}$$

The only difference between Equation 2 and Equation 4 is the presence of the normalization factor in the latter, that is: $\log \sum_{y'}^{|\mathcal{Y}|} \mathrm{P}_{\mathcal{B}}(y', \mathbf{x}^{(j)})$. Due to this normalization, the values of θ maximizing Equation 4 are not the same as those that maximize Equation 2. We provide two intuitions about "why maximizing the CLL would provide a better model of the conditional distribution":

1. It allows the parameters to be set in such a way as to reduce the effect of the conditional attribute independence assumption that is present in the BN structure and that might be violated in data.
2. We have $\mathrm{LL}(\mathcal{B}) = \mathrm{CLL}(\mathcal{B}) + \mathrm{LL}(\mathcal{B}\backslash y)$. If optimizing $\mathrm{LL}(\mathcal{B})$, most of the attention will be given to $\mathrm{LL}(\mathcal{B}\backslash y)$ – because $\mathrm{CLL}(\mathcal{B}) \ll \mathrm{LL}(\mathcal{B}\backslash y)$ – which will often lead to poor estimates for classification.

Note, that if the structure is correct, maximizing both LL and CLL should lead to the same results [20]. There is unfortunately no closed-form solution for θ such that the CLL would be maximized; we thus have to resort to numerical optimization methods over the space of parameters.

Like any Bayesian network model, a class-conditional BN model is composed of a graphical structure and of parameters ($\boldsymbol{\theta}$) quantifying the dependencies in the structure. For any BN $\mathcal{B}$, the corresponding CCBN will be based on graph $\mathcal{B}^*$ (where $\mathcal{B}^*$ is a sub-graph of $\mathcal{B}$) whose parameters are optimized by maximizing the CLL. We present below a slightly rephrased definition from [19]:

Definition 1 A class-conditional Bayesian network model $\mathcal{M}^{\mathcal{B}^*}$ is the set of conditional distributions based on the network $\mathcal{B}^*$ equipped with any strictly positive parameter set $\boldsymbol{\theta}^{\mathcal{B}^*}$; that is the set of all functions from $(X_1, X_2, ..., X_n)$ to a distribution on Y takes the form of Equation 1.

This means that the nodes in $\mathcal{B}^*$ are nodes comprising only the Markov blanket of the class y. However, in most cases, for BN classifiers, a structure is learned without the class attribute and, afterwards, class is added as the parent of all the attributes. Second, class does not take any other node as its parents. This has the effect that each attribute is in the Markov blanket of the class. We will assume that the parents of class attribute constitute an empty set and, therefore, replace parameters characterizing the class attribute from $\theta_{y^{(j)}|\Pi_0(\mathbf{x}^{(j)})}$ with $\theta_{y^{(j)}}$. We will also drop the superscript j in equations for clarity.

3 Parameterization 1: Discriminative CCBN Model

Logistic regression (LR) is the CCBN model associated to the NB structure optimizing Equation 1. Typically, LR learns a weight for each attribute-value (per-class). However, one can extend LR by considering all or some subset of possible quadratic, cubic, or higher-order features [12,25]. We define discriminative CCBN as:

Definition 2 A discriminative class-conditional Bayesian Network model $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ is a CCBN such that Equation 1 is re-parameterized in form of parameter $\boldsymbol{\beta}$ such that $\boldsymbol{\beta} = \log \boldsymbol{\theta}$ and parameter $\boldsymbol{\beta}$ is obtained by maximizing the CLL.

Let us re-define $\mathrm{P}_{\mathcal{B}}(y|\mathbf{x})$ in Equation 4 and write it on a per datum basis as:

$$\mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = \frac{\exp(\log \theta_y + \sum_{i=1}^{n} \log \theta_{x_i|y,\Pi_i(\mathbf{x})})}{\sum_{y'}^{|\mathcal{Y}|} \exp(\log \theta_{y'} + \sum_{i=1}^{n} \log \theta_{x_i|y',\Pi_i(\mathbf{x})})}. \tag{5}$$

In light of Definition 2, let us define a parameter $\beta_\bullet$ that is associated with each parameter $\theta_\bullet$ in Equation 5, such that:

$$\log \theta_y = \beta_y, \quad \text{and} \quad \log \theta_{x_i|y,\Pi_i(\mathbf{x})} = \beta_{y,x_i,\Pi_i}.$$

Now Equation 5 can be written as:

$$\mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = \frac{\exp(\beta_y + \sum_{i=1}^{n} \beta_{y,x_i,\Pi_i})}{\sum_{y'=1}^{|\mathcal{Y}|} \exp(\sum_{y'} \beta_{y'} + \sum_{i=1}^{n} \beta_{y',x_i,\Pi_i})}. \tag{6}$$

One can see that this has led to the logistic function of the form $\frac{1}{1+\exp(-\beta^T \mathbf{x})}$ for binary classification and softmax $\frac{\exp(-{\beta_{\mathbf{y}}}^T \mathbf{x})}{\sum'_y(\exp(-{\beta_{\mathbf{y}'}}^T \mathbf{x}))}$ for multi-class classification. Such a formulation is a Logistic Regression classifier. Therefore, we can state that a discriminative CCBN model with naive Bayes structure is a (vanilla) logistic regression classifier.

In light of Definition 2, CLL optimized by $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$, on a per-datum-basis, can be specified as:

$$\log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = (\beta_y + \sum_{i=1}^{n} \beta_{y,x_i,\Pi_i}) - \log(\sum_{y'=1}^{|\mathcal{Y}|} \exp(\beta_{y'} + \sum_{i=1}^{n} \beta_{y',x_i,\Pi_i})). \tag{7}$$

Now, we will have to rely on an iterative optimization procedure based on gradient-descent. Therefore, let us first calculate the gradient of parameters in the model. The gradient of the parameters in Equation 7 can be computed as:

$$\frac{\partial \log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x})}{\partial \beta_{y:k}} = (\mathbf{1}_{y=k} - \mathrm{P}(k|\mathbf{x}))\,, \tag{8}$$

for the class parameters. For the other parameters, we can compute the gradient as:

$$\frac{\partial \log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x})}{\partial \beta_{y:k,x_i:j,\Pi_i:l}} = (\mathbf{1}_{y=k} - \mathrm{P}(k|\mathbf{x}))\, \mathbf{1}_{x_i=j} \mathbf{1}_{\Pi_i=l}, \tag{9}$$

where $\mathbf{1}$ is the indicator function. Note, that we have used the notation $\beta_{y:k,x_i:j,\Pi_i:l}$ to denote that class y has the value k, attribute x_i has the value j and its parents (Π_i) have the value l. If the attribute has multiple parent attributes, then l represents a combination of parent attribute values.

4 Parameterization 2: Extended CCBN Model

The name *Extended CCBN Model* is inspired from [8], where the method named Extended Logistic Regression (ELR) is proposed. ELR is aimed at extending LR and leads to discriminative training of BN parameters. We define:

Definition 3 [8] – An extended class-conditional Bayesian Network model $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ is a CCBN such that the parameters ($\boldsymbol{\theta}$) satisfy the constraints in Equation 3 and is obtained by maximizing the CLL in Equation 4.

Let us re-define $\mathrm{P}_{\mathcal{B}}(y|\mathbf{x})$ in Equation 4 on a per-datum-basis as:

$$\log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = (\log \theta_y + \sum_{i=1}^{n} \log \theta_{x_i|y,\Pi_i(\mathbf{x})}) - \log \sum_{y'}^{|\mathcal{Y}|} (\theta_{y'} \prod_{i=1}^{n} \theta_{x_i|y',\Pi_i(\mathbf{x})}). \tag{10}$$

Let us consider the case of optimizing parameters associated with the attributes $\theta_{x_i|y,\Pi_i(\mathbf{x})}$. Parameters associated with the class can be obtained similarly. We will re-write $\theta_{x_i|y,\Pi_i(\mathbf{x})}$ as $\theta_{x_i:j|y:k,\Pi_i:l}$ which represents attribute i (x_i) taking value j, class (y) taking value k and its parents (Π_i) takes value l. Now we can write the gradient as:

$$\begin{aligned}\frac{\partial \log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x})}{\partial \theta_{x_i:j'|y:k,\Pi_i:l}} &= \left(\frac{\mathbf{1}_{y=k}\mathbf{1}_{x_i=j'}\mathbf{1}_{\Pi_i=l}}{\theta_{x_i:j'|y:k,\Pi_i:l}} - \frac{\hat{\mathrm{P}}(k|\mathbf{x})\mathbf{1}_{x_i=j'}\mathbf{1}_{\Pi_i=l}}{\theta_{x_i:j'|y:k,\Pi_i:l}} \right), \\ &= \frac{\mathbf{1}_{x_i=j'}\mathbf{1}_{\Pi_i=l}}{\theta_{x_i:j'|y:k,\Pi_i:l}} \left(\mathbf{1}_{y=k} - \hat{\mathrm{P}}(k|\mathbf{x}) \right).\end{aligned}$$

Enforcing constraints that $\sum_{j'} \theta_{x_i:j'|y:k,\Pi_i:l} = 1$, we introduce a new parameters β and re-parameterize as:

$$\theta_{x_i:j'|y:k,\Pi_i:l} = \frac{\exp(\beta_{x_i:j'|y:k,\Pi_i:l})}{\sum_{j''} \exp(\beta_{x_i:j''|y:k,\Pi_i:l})}. \tag{11}$$

It will be helpful if we differentiate $\theta_{x_i:j'|y:k,\Pi_i:l}$ with respect to $\beta_{x_i:j|y:k,\Pi_i:l}$ (the use of notation j and j' will become obvious when we apply the chain rule afterwards), we get:

$$\begin{aligned}\frac{\partial \theta_{x_i:j'|y:k,\Pi_i:l}}{\partial \beta_{x_i:j|y:k,\Pi_i:l}} &= \frac{\exp(\beta_{x_i:j'|y:k,\Pi_i:l})\mathbf{1}_{y=k}\mathbf{1}_{x_i=j'=j}\mathbf{1}_{\Pi_i=l}}{\sum_{j''} \exp(\beta_{x_i:j''|y:k,\Pi_i:l})} \\ &- \frac{\exp(\beta_{x_i:j'|y:k,\Pi_i:l}) \exp(\beta_{x_i:j''|y:k,\Pi_i:l})\mathbf{1}_{x_i=j''=j}\mathbf{1}_{\Pi_i=l}}{\left(\sum_{j''} \exp(\beta_{x_i:j''|y:k,\Pi_i:l})\right)^2}, \\ &= \mathbf{1}_{y=k}\mathbf{1}_{x_i=j'=j}\mathbf{1}_{\Pi_i=l}\theta_{x_i:j|y:k,\Pi_i:l} - \\ &\quad \mathbf{1}_{x_i=j''=j}\mathbf{1}_{\Pi_i=l}\theta_{x_i:j'|y:k,\Pi_i:l}\theta_{x_i:j|y:k,\Pi_i:l}, \\ &= (\mathbf{1}_{y=k} - \theta_{x_i:j|y:k,\Pi_i:l})\mathbf{1}_{x_i=j}\mathbf{1}_{\Pi_i=l}\theta_{x_i:j'|y:k,\Pi_i:l}.\end{aligned}$$

Applying the chain rule:

$$\begin{aligned}\frac{\partial \log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x})}{\partial \beta_{x_i:j|y:k,\Pi_i:l}} &= \sum_{j'} \frac{\partial \log \mathrm{P}(y|\mathbf{x})}{\partial \theta_{x_i:j'|y:k,\Pi_i:l}} \frac{\partial \theta_{x_i:j'|y:k,\Pi_i:l}}{\partial \beta_{x_i:j|y:k,\Pi_i:l}}, \\ &= (\mathbf{1}_{y=k}\mathbf{1}_{x_i=j}\mathbf{1}_{\Pi_i=l} - \mathbf{1}_{x_i=j}\mathbf{1}_{\Pi_i=l}\mathrm{P}(k|\mathbf{x})) - \\ \theta_{x_i:j|y:k,\Pi_i:l} &\sum_{j'} \left(\mathbf{1}_{y=k}\mathbf{1}_{x_i=j'}\mathbf{1}_{\Pi_i=l} - \mathbf{1}_{x_i=j'}\mathbf{1}_{\Pi_i=l}\mathrm{P}(k|\mathbf{x})\right),\end{aligned} \tag{12}$$

we get the gradient of $\log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x})$ with respect to parameter $\beta_{x_i:j|y:k,\Pi_i:l}$. Now one can use the transformation of Equation 11 to obtain the desired parameters of extended CCBN. Note that Equation 12 corresponds to Equation 9. The only difference is the presence of the normalization term that is subtracted from the gradient in Equation 12.

5 Parameterization 3: Combined generative/discriminative parameterization: Weighted CCBN Model

We define a weighted CCBN model as follows:

Definition 4 A weighted conditional Bayesian Network model $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ is a CCBN such that Equation 1 has an extra weight parameter associated with every θ such that it is re-parameterized as: $\boldsymbol{\theta}^{\mathbf{w}}$, where parameter $\boldsymbol{\theta}$ is learned by optimizing the LL and parameter $\mathbf{w}$ is obtained by maximizing the CLL.

In light of Definition 4, let us re-define Equation 1 to incorporate weights as:

$$\mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = \frac{\theta_y^{w_y} \prod_{i=1}^{n} \theta_{x_i|y,\Pi_i(\mathbf{x})}^{w_{y,x_i,\Pi_i}}}{\sum_{y'}^{|\mathcal{Y}|} \theta_{y'}^{w_{y'}} \prod_{i=1}^{n} \theta_{x_i|y',\Pi_i(\mathbf{x})}^{w_{y',x_i,\Pi_i}}}. \tag{13}$$

The corresponding weighted CLL can be written as:

$$\log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = (w_y \log \theta_y + \sum_{i=1}^{n} w_{y,x_i,\Pi_i} \log \theta_{x_i|y,\Pi_i(\mathbf{x})}) - \log \sum_{y'}^{|\mathcal{Y}|} (\theta_{y'}^{w_y} \prod_{i=1}^{n} \theta_{x_i|y',\Pi_i(\mathbf{x})}^{w_{y,x_i,\Pi_i}}). \tag{14}$$

Note, that Equation 14 is similar to Equation 10 except for the introduction of weight parameters. The flexibility to learn parameter $\boldsymbol{\theta}$ in a prior generative process of learning greatly simplifies subsequent calculations of $\mathbf{w}$ in a discriminative search. Since $\mathbf{w}$ is a free-parameter and there is no sum-to-one constraint, its optimization is simpler than for $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$. The gradient of the parameters in Equation 14 can be computed as:

$$\frac{\partial \log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x})}{\partial w_{y:k}} = (\mathbf{1}_{y=k} - \mathrm{P}(k|\mathbf{x})) \log \theta_{y|\Pi_0(\mathbf{x})}, \tag{15}$$

for the class y, while for the other parameters:

$$\frac{\partial \log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x})}{\partial w_{y:k,x_i:j,\Pi_i:l}} = (\mathbf{1}_{y=k} - \mathrm{P}(k|\mathbf{x}))\, \mathbf{1}_{x_i=j} \mathbf{1}_{\Pi_i=l} \log \theta_{x_i|y,\Pi_i(\mathbf{x})}. \tag{16}$$

One can see that Equations 15 and 16 correspond to Equations 8 and 9. The only difference between them is the presence of the $\log \theta_{\bullet}$ factor in the $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ case.

A brief summary of these parameterizations is also given in Table 1.

	Generative – Maximize LL	**Discriminative** – Maximize CLL		
		Extended CCBN model	**Discriminative CCBN model**	**Weighted CCBN model**
Description	Estimate parameters of joint-distribution $\mathrm{P}(y, \mathbf{x})$	Estimate parameters of class-conditional distribution $\mathrm{P}(y \mid \mathbf{x})$	Estimate parameters of class-conditional distribution $\mathrm{P}(y \mid \mathbf{x})$	Estimate parameters of class-conditional distribution $\mathrm{P}(y \mid \mathbf{x})$
BN structure	$\mathcal{B}$	$\mathcal{B}^*$	$\mathcal{B}^*$	$\mathcal{B}^*$
CCBN model	Not applicable	$\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$	$\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$	$\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$
Form	$\mathrm{P}(y, \mathbf{x} \mid \boldsymbol{\theta})$	$\mathrm{P}(y \mid \mathbf{x}, \boldsymbol{\theta})$	$\mathrm{P}(y \mid \mathbf{x}, \boldsymbol{\beta})$	$\mathrm{P}(y \mid \mathbf{x}, \boldsymbol{\theta}, \mathbf{w})$
Formula	$\theta_y \prod_{i=1}^{n} \theta_{x_i \mid y, \Pi_i(\mathbf{x})}$	$\frac{\theta_y \prod_{i=1}^{n} \theta_{x_i \mid y, \Pi_i(\mathbf{x})}}{\sum_{y'}^{\mid \mathcal{Y} \mid} \theta_{y'} \prod_{i=1}^{n} \theta_{x_i \mid y', \Pi_i(\mathbf{x})}}$	$\frac{\exp(\beta_y + \sum_{i=1}^{n} \beta_{y, x_i, \Pi_i})}{\sum_{y'=1}^{\mid \mathcal{Y} \mid} \exp(\sum_{y'} \beta_{y'} + \sum_{i=1}^{n} \beta_{y', x_i, \Pi_i})}$	$\frac{\theta_y^{w_y} \prod_{i=1}^{n} \theta_{x_i \mid y, \Pi_i(\mathbf{x})}^{w_{y, x_i, \Pi_i}}}{\sum_{y'}^{\mid \mathcal{Y} \mid} \theta_{y'}^{w_{y'}} \prod_{i=1}^{n} \theta_{x_i \mid y', \Pi_i(\mathbf{x})}^{w_{y', x_i, \Pi_i}}}$
Constraints	$\pi \in [0,1]^{\mid \mathcal{Y} \mid}$, $\theta_{i,y} \in [0,1]^{\mid \mathcal{X}_i \mid}$	$\pi \in [0,1]^{\mid \mathcal{Y} \mid}$, $\theta_{i,y} \in [0,1]^{\mid \mathcal{X}_i \mid}$	$\alpha_1 = 0, \forall_i \ \beta_{i,1} = 0$	$\pi \in [0,1]^{\mid \mathcal{Y} \mid}$, $\theta_{i,y} \in [0,1]^{\mid \mathcal{X}_i \mid}$
Optimized Param.	$\boldsymbol{\theta}$	$\boldsymbol{\theta}$	$\boldsymbol{\beta}$	$\mathbf{w}$
'Fixed' Param.	None	None	None	$\boldsymbol{\theta}$
Note	Not applicable	Not applicable	Not applicable	π and Θ are fixed to their MAP estimates

Table 1: Comparison of different parameter learning techniques for Bayesian Network Classifiers.

5.1 Combined Discriminative/Generative Regularization

Contrary to $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$, $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ parameterization offers an elegant framework for blending discriminative and generative learned parameters. With regularization, one can interpolate between the two paradigms. For example, let us modify Equation 13 as:

$$\mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = \frac{1}{\mathcal{Z}} \exp(w_y \log \theta_y + \sum_{i=1}^{n} w_{y,x_i,\Pi_i} \log \theta_{x_i|y,\Pi_i(\mathbf{x})}) + \frac{\lambda}{2}\|\mathbf{w}\|^2,$$

where $\mathcal{Z}$ is the normalization constant and λ is the parameter controlling regularization. The new term will penalize large (and heterogeneous) parameter values. Larger λ values will cause the classifier to progressively ignore the data and assign more uniform class probabilities. Alternatively one could penalize deviations from the BN conditional independence assumption by centering the regularization term at one rather than zero:

$$\mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = \frac{1}{\mathcal{Z}} \exp(w_y \log \theta_y + \sum_{i=1}^{n} w_{y,x_i,\Pi_i} \log \theta_{x_i|y,\Pi_i(\mathbf{x})}) + \frac{\lambda}{2}\|\mathbf{w} - \mathbf{1}\|^2.$$

Doing so allows the regularization parameter λ to be used to interpolate between the generative model and the discriminative model.

5.2 On Initialization of Parameters

Initialization of the parameters, which sets the starting point for the optimization, is an orthogonal element to the speed of convergence that this paper addresses. Obviously, a better starting point (in terms of CLL), will make the optimization easier and conversely. In this paper, we will study two different starting points for the parameters:

Zeros This is the standard initialization where all the optimized parameters are initialized with 0 [18].

Generative estimates Given that our approach utilizes generative estimates, a fair comparison with other approaches should study starting from the generative estimates for all approaches. This will correspond to the θs initialized to the generative estimates for Parameterizations 1 and 2, and to the ws initialized to one for Parameterization 3.

Note that in the "Zeros" case, only our proposed Weighted CCBN parametrization requires a first (extra) pass over the dataset to compute the generative estimates, while for the "Generative estimates" case all methods require this pass (when we report training time, we always report the full training time).

6 Related Work

There have been several comparative studies of discriminative and generative structure and parameter learning of Bayesian Networks [7,9,15]. In all these works, generative parameter training is the estimation of parameters based on empirical estimates whereas discriminative training of parameters is actually the estimation of the parameters of CCBN models such as $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ or $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$. The $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ model was first proposed in [7]. Our work differs from these previous works as our goal is to highlight different parameterization of CCBN models and investigate their inter-relationship. Particularly, we are interested in the learning of parameters corresponding to a weighted CCBN model.

An approach for discriminative learning of the parameters of BN based on discriminative computation of frequencies from the data is presented in [21]. *Discriminative Frequency Estimates* (DFE) are computed by injecting a discriminative element to generative computation of the probabilities. During the frequencies computation process, rather than updating the count tables as individual datum arrives, DFE estimates how well the current classifier does on the arriving data point and then update the tables only in proportion to the classifier's performance. For example, they propose a simple error measure, as: $L(\mathbf{x}) = \mathrm{P}(y|\mathbf{x}) - \hat{\mathrm{P}}(y|\mathbf{x})$, where $\mathrm{P}(y|\mathbf{x})$ is the true probability of class y given the datum $\mathbf{x}$, and $\hat{\mathrm{P}}(y|\mathbf{x})$ is the predicted probability. The counts are updated as: $\theta_{ijk}^{t+1} = \theta_{ijk}^{t} + L(\mathbf{x})$. Several iterations over the dataset are required. The algorithm is inspired from Perceptron based training and is shown to be an effective discriminative parameter learning approach.

7 Empirical Results

In this section, we compare and analyze the performance of our proposed algorithms and related methods on 72 natural domains from the UCI repository of machine learning [5]. The experiments are conducted on the datasets described in Table 2.

There are a total of 72 datasets, 41 datasets with less than 1000 instances, 21 datasets with between 1000 and 10000 instances, and 11 datasets with more than 10000 instances. Each algorithm is tested on each dataset using 5 rounds of 2-fold cross validation. 2-fold cross validation is used in order to maximize the variation in the training data from trial to trial, which is advantageous when estimating bias and variance. Note that the source code with running instructions is provided as a supplementary material to this paper.

We compare four metrics: 0-1 Loss, RMSE, Bias and Variance. We report Win-Draw-Loss (W-D-L) results when comparing the 0-1 Loss, RMSE, bias and variance of two models. A two-tail binomial sign test is used to determine the significance of the results. Results are considered significant if $p \leq 0.05$. We report results on two categories of datasets. The first category, labeled *All*, consists of all datasets in Table 2. The second category, labeled *Big*, consists of datasets that have more than 10000 instances. Numeric attributes are

Domain	Case	Att	Class	Domain	Case	Att	Class
Poker-hand	1175067	11	10	Annealing	898	39	6
Covertype	581012	55	7	Vehicle	846	19	4
Census-Income(KDD)	299285	40	2	PimaIndiansDiabetes	768	9	2
Localization	164860	7	3	BreastCancer(Wisconsin)	699	10	2
Connect-4Opening	67557	43	3	CreditScreening	690	16	2
Statlog(Shuttle)	58000	10	7	BalanceScale	625	5	3
Adult	48842	15	2	Syncon	600	61	6
LetterRecognition	20000	17	26	Chess	551	40	2
MAGICGammaTelescope	19020	11	2	Cylinder	540	40	2
Nursery	12960	9	5	Musk1	476	167	2
Sign	12546	9	3	HouseVotes84	435	17	2
PenDigits	10992	17	10	HorseColic	368	22	2
Thyroid	9169	30	20	Dermatology	366	35	6
Pioneer	9150	37	57	Ionosphere	351	35	2
Mushrooms	8124	23	2	LiverDisorders(Bupa)	345	7	2
Musk2	6598	167	2	PrimaryTumor	339	18	22
Satellite	6435	37	6	Haberman'sSurvival	306	4	2
OpticalDigits	5620	49	10	HeartDisease(Cleveland)	303	14	2
PageBlocksClassification	5473	11	5	Hungarian	294	14	2
Wall-following	5456	25	4	Audiology	226	70	24
Nettalk(Phoneme)	5438	8	52	New-Thyroid	215	6	3
Waveform-5000	5000	41	3	GlassIdentification	214	10	3
Spambase	4601	58	2	SonarClassification	208	61	2
Abalone	4177	9	3	AutoImports	205	26	7
Hypothyroid(Garavan)	3772	30	4	WineRecognition	178	14	3
Sick-euthyroid	3772	30	2	Hepatitis	155	20	2
King-rook-vs-king-pawn	3196	37	2	TeachingAssistantEvaluation	151	6	3
Splice-junctionGeneSequences	3190	62	3	IrisClassification	150	5	3
Segment	2310	20	7	Lymphography	148	19	4
CarEvaluation	1728	8	4	Echocardiogram	131	7	2
Volcanoes	1520	4	4	PromoterGeneSequences	106	58	2
Yeast	1484	9	10	Zoo	101	17	7
ContraceptiveMethodChoice	1473	10	3	PostoperativePatient	90	9	3
German	1000	21	2	LaborNegotiations	57	17	2
LED	1000	8	10	LungCancer	32	57	3
Vowel	990	14	11	Contact-lenses	24	5	3
Tic-Tac-ToeEndgame	958	10	2				

Table 2: Details of Datasets (UCI Domains)

discretized by using the Minimum Description Length (MDL) discretization method [4]. A missing value is treated as a separate attribute value and taken into account exactly like other values. Optimization is done with L-BFGS [2][1] [2].

We experiment with three Bayesian network structures that is: naive Bayes (NB), Tree-Augmented naive Bayes (TAN) and k-Dependence Bayesian Network (KDB) with $K = 1$. We denote $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$, $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ with naive Bayes structure as NB^{w}, NB^{d} and NB^{e} respectively. With TAN structure, $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$, $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ are denoted as $\mathrm{TAN}^{\mathrm{w}}$, $\mathrm{TAN}^{\mathrm{d}}$ and $\mathrm{TAN}^{\mathrm{e}}$. With KDB ($K = 1$), $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$, $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ are denoted as KDB-1^{w}, KDB-1^{d} and KDB-1^{e}.

As discussed in Section 5.2, we initialize the parameters to the log of the MAP estimates (or parameters optimized by generative learning). The '(I)' in

[1] The algorithm terminates when improvement in the objective function, given by $\frac{(f_t - f_{t+1})}{\max\{|f_t|,|f_{t+1}|,1\}}$, drops below 10^{-32}, or the no. of iterations exceeds 10000 [26].

[2] The original L-BFGS implementation of [2] from `http://users.eecs.northwestern.edu/~nocedal/lbfgsb.html` is used.

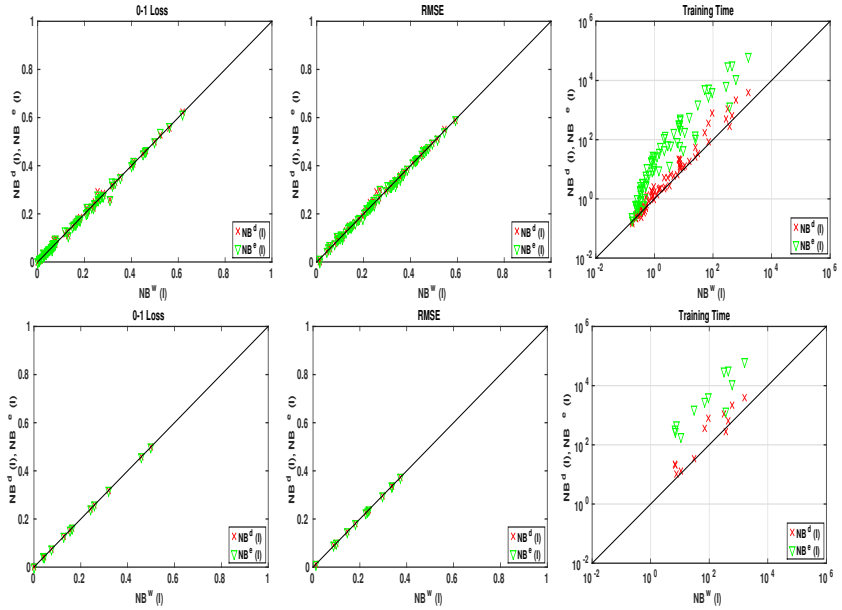

Fig. 1: Comparative scatter of No. of iterations (left), training time (middle) and RMSE (right) for NB^w, NB^d and NB^e on *All* datasets (Top Row) and on *Big* datasets (Bottom Row). NB^w on the X-axis, NB^d (red-cross) and NB^e (green-triangle) on the Y-axis.

the label represents this initialization. An absence of '(I)' means the parameters are initialized to zero.

7.1 NB Structure

Comparative scatter plots on all 72 datasets for 0-1 Loss, RMSE and training time values for NB^w, NB^d and NB^e are shown in Figure 1. Training time plots are on the log scale. The plots are shown separately for *Big* datasets. It can be seen that the three parameterizations have a similar spread of RMSE values, however, NB^w is greatly advantaged in terms of its training time. This computational advantage arises from the relative convergence properties of the three parameterizations and is discussed in Section 7.4. Given that NB^w achieves equivalent accuracy with much less computation indicates that it is a more effective parameterization than NB^d and NB^e.

The geometric means of the 0-1 Loss, RMSE and training time results are shown in Figure 2. Note, results are normalized with respect to NB, and, therefore, NB averaged results are all 1. They are also plotted on the graph for reference. Note, training time results are in log-scale. It can be seen that discriminative methods on *Big* datasets are an order of magnitude better than generative learning in terms of 0-1 Loss. Though discriminative training results in greatly improved 0-1 Loss and RMSE error, this gain in accuracy comes at a considerable cost in training time.

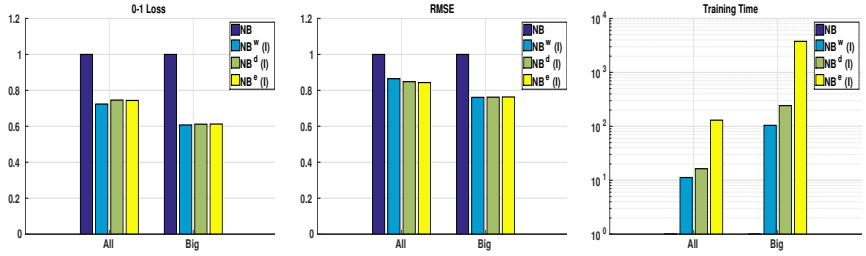

Fig. 2: Geometric mean of No. of iterations, training time and RMSE for NB, NB^w, NB^d and NB^e on *All* and *Big* datasets. Results are normalized with respect to NB.

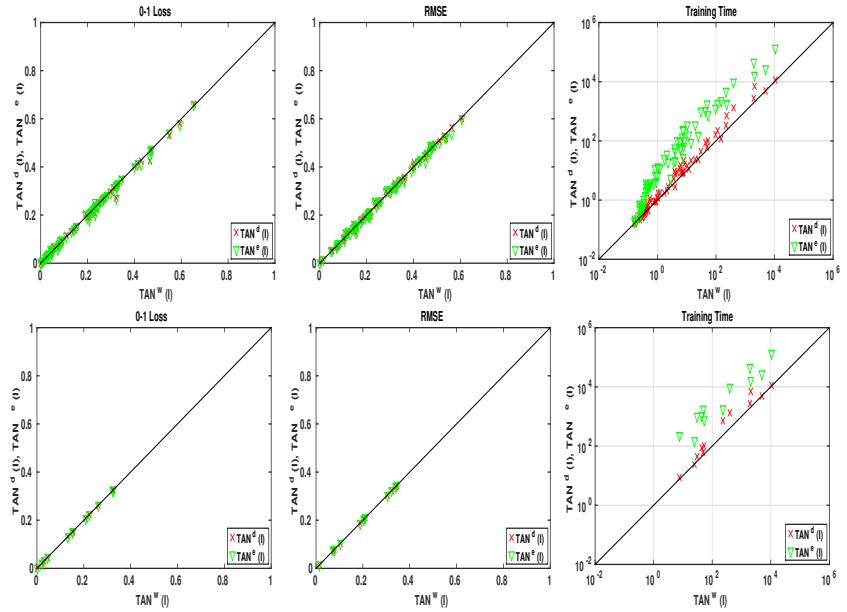

Fig. 3: Comparative scatter of No. of iterations (left), training time (middle) and RMSE (right) for TAN^w, TAN^d and TAN^e on *All* datasets (Top Row) and on *Big* datasets (Bottom Row). TAN^w on the X-axis, TAN^d (red-cross) and TAN^e (green-triangle) on the Y-axis.

7.2 TAN Structure

Figure 3 shows the comparative spread of 0-1 Loss, RMSE and training time of TAN^w, TAN^d and TAN^e on *All* and *Big* datasets. A trend similar to that of NB can be seen. With a similar spread of 0-1 Loss and RMSE among the three parameterizations, training time is greatly improved for TAN^w when compared with TAN^d and TAN^e. Geometric average of the 0-1 Loss, RMSE and training time results are shown in Figure 4. Note, results are normalized with respect to TAN, and, therefore, TAN averaged results are all equal to 1. They are also plotted on the graph for reference.

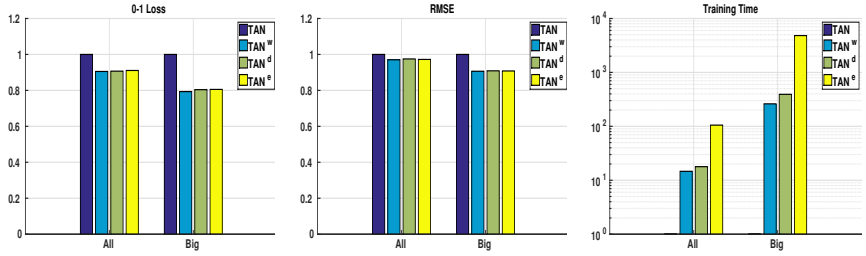

Fig. 4: Geometric mean of No. of iterations, training time and RMSE for TAN$^{\mathrm{w}}$, TAN$^{\mathrm{d}}$ and TAN$^{\mathrm{e}}$ on *All* and *Big* datasets. Results are normalized with respect to TAN.

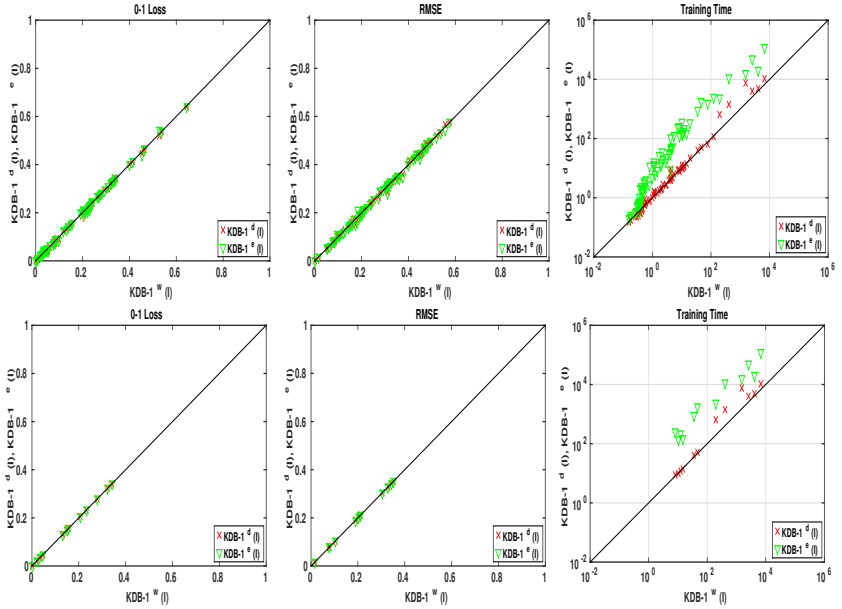

Fig. 5: Comparative scatter of No. of iterations (left), training time (middle) and RMSE (right) for KDB-1$^{\mathrm{w}}$, KDB-1$^{\mathrm{d}}$ and KDB-1$^{\mathrm{e}}$ on *All* datasets (Top Row) and on *Big* datasets (Bottom Row). KDB-1$^{\mathrm{w}}$ on the X-axis, KDB-1$^{\mathrm{d}}$ (red-cross) and KDB-1$^{\mathrm{e}}$ (green-triangle) on the Y-axis.

7.3 KDB ($K = 1$) Structure

Figure 5 shows the comparative spread of 0-1 Loss, RMSE and training time of KDB-1$^{\mathrm{w}}$, KDB-1$^{\mathrm{d}}$ and KDB-1$^{\mathrm{e}}$ on *All* and *Big* datasets. Like NB and TAN, it can be seen that a similar spread of 0-1 Loss and RMSE is present among the three parameterizations of discriminative learning. Similarly, training time is greatly improved for KDB-1$^{\mathrm{w}}$ when compared with KDB-1$^{\mathrm{d}}$ and KDB-1$^{\mathrm{e}}$. Geometric average of the 0-1 Loss, RMSE and training time results are shown in Figure 6. Note, results are normalized with respect to KDB ($K = 1$), and,

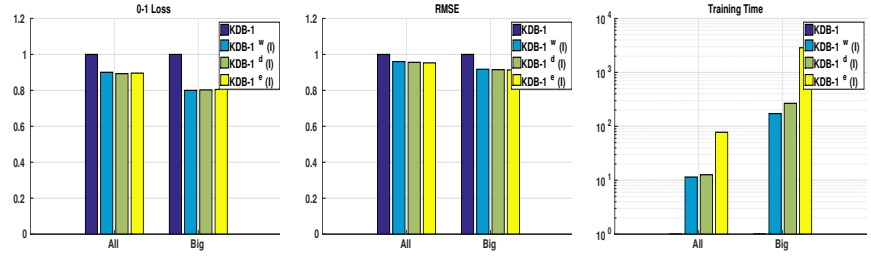

Fig. 6: Geometric mean of No. of iterations, training time and RMSE for KDB-1$^{\mathrm{w}}$, KDB-1$^{\mathrm{d}}$ and KDB-1$^{\mathrm{e}}$ on *All* and *Big* datasets. Results are normalized with respect to KDB-1.

therefore, KDB ($K = 1$) averaged results are all equal to 1. They are also plotted on the graph for reference.

7.4 Convergence Analysis

A comparison of the convergence of Negative Log-Likelihood (NLL) of the three parameterizations on some sample datasets with NB, TAN and KDB ($K = 1$) structure is shown in Figure 7 and 8. In Figure 7, parameters are initialized to zero, whereas, in Figure 8, parameters are initialized to the log of the MAP estimates. It can be seen that for all three structures and for both initializations, $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ not only converges faster but also reaches its asymptotic value much quicker than the $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$. The same trend was observed on all 73 datasets.

To quantify how much $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ is faster than the other two parameterizations, we plot a histogram of the number of iterations it takes $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ after five iterations to reach the negative log-likelihood that $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ achieved at fifth iteration. If the three parameterizations follow similar convergence, one should expect many zeros in the histogram. Note that if after fifth iteration, NLL of $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ is greater than that of $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$, we we plot the negative of the number of iterations it takes $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ to reach the NLL of $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$. Similarly, if after fifth iteration, NLL of $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ is greater than that of $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$, we we plot the negative of the number of iterations it takes $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ to reach the NLL of $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$. Figures 9, 10 and 11 show these histogram plots for NB, TAN and KDB ($K = 1$) structure respectively. It can be seen that $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ (with all three structures) achieves a NLL that otherwise, will take on average 10 more iterations over the data for $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ and 15 more iterations for $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$. This is an extremely useful property of $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ especially for big data where iterating through the dataset is expensive, but the more complex network structures are difficult to optimize.

7.5 Comparison with MAP

The purpose of this section is to compare the performance of the discriminative learning with that of generative learning. In Table 3, we compare the

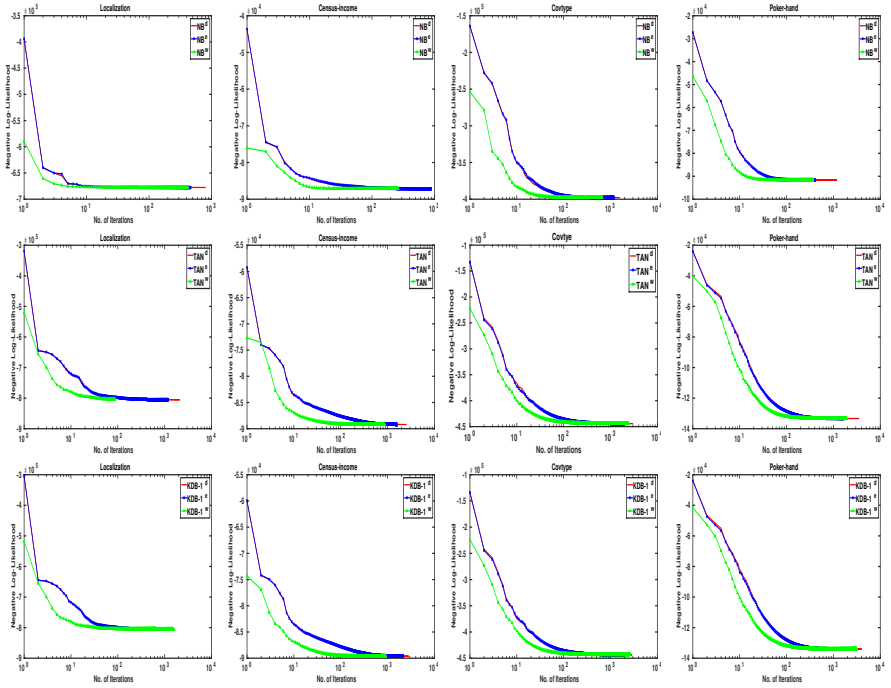

Fig. 7: Comparison of rate of convergence on the four biggest datasets for NB (top row), TAN (middle row) and KDB ($K = 1$) (bottom row) structures. The X-axis is on log scale. Parameters are initialized to zero.

performance of NB^{w} with NB (i.e., naive Bayes with MAP estimates of probabilities), TAN^{w} with TAN (i.e., TAN with MAP estimates of probabilities) and KDB-1^{w} with KDB ($K = 1$) (i.e., KDB with MAP estimates of probabilities). We use NB^{w}, TAN^{w} and KDB-1^{w} as a representative of discriminative learning - since $\mathcal{M}_{\text{w}}^{\mathcal{B}^*}$, $\mathcal{M}_{\text{d}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\text{e}}^{\mathcal{B}^*}$ have similar 0-1 Loss and RMSE profile. It can be see that the discriminative learning of parameters has significantly lower bias but higher variance. On big datasets, it can be seen that discriminative learning results in much better 0-1 Loss and RMSE performance.

Note that though discriminative learning (optimizes the parameters characterizing CCBN) has better 0-1 Loss and RMSE performance than generative learning (optimizing joint probability), – generative learning has the advantage of being extremely fast as it incorporates counting of sufficient statistics from the data. Another advantage of generative learning is its capability of back-off in case a certain combination does not exist in the data. For example, TAN and KDB classifiers if have not encountered a $<$ `feature-value, parent-value, class-value` $>$ combination at training time can resort back to $<$ `feature-value, class-value` $>$ at testing time. For example TAN classifier can resort back to NB and NB can resort back to class prior probabilities. Such elegantly back-tracking is missing from discrimi-

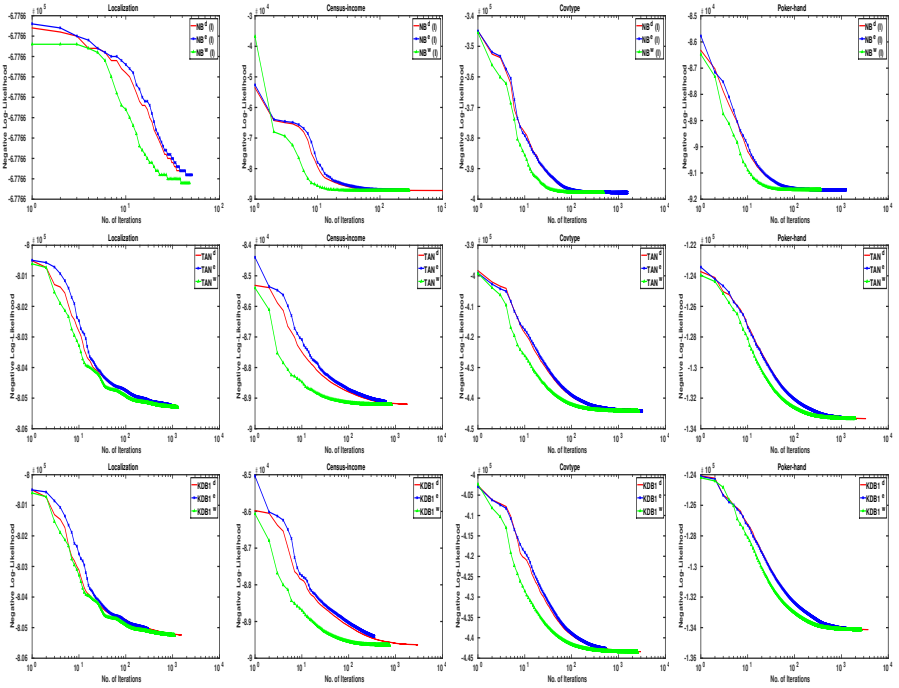

Fig. 8: Comparison of rate of convergence on the four biggest datasets for NB (top row), TAN (middle row) and KDB ($K = 1$) (bottom row) structures. The X-axis is on log scale. Parameters are initialized to the log of the MAP estimates.

	NB$^{\mathbf{w}}$ vs. NB		**TAN$^{\mathbf{w}}$ vs. TAN**		**KDB-1$^{\mathbf{w}}$ vs. KDB-1**	
	W-D-L	p	W-D-L	p	W-D-L	p
All Datasets						
Bias	62/3/7	**<0.001**	50/4/18	**<0.001**	54/5/13	**<0.001**
Variance	19/3/50	**<0.001**	21/2/49	0.011	19/4/49	**<0.001**
0-1 Loss	45/4/23	0.010	34/3/35	1	39/4/29	0.275
RMSE	45/3/24	0.015	25/1/46	0.017	29/2/41	0.1882
Big Datasets						
0-1 Loss	11/1/0	**<0.001**	11/1/0	**<0.001**	11/0/1	**<0.001**
RMSE	11/0/1	**<0.001**	11/0/1	**<0.001**	11/0/1	**<0.001**

Table 3: Win-Draw-Loss: NB$^{\mathrm{w}}$ vs. NB, TAN$^{\mathrm{w}}$ vs. TAN and KDB-1$^{\mathrm{w}}$ vs. KDB-1. Significant results are shown in bold.

native learning. If a certain combination does not exist in the data, parameters associated to that parameter will not be optimized and will remain fixed to the initialized value (for example 0). A discriminative classifier will have no way of handling unseen combinations but to ignore the combination if that combination occurs in the testing data. How to incorporate such hierarchical

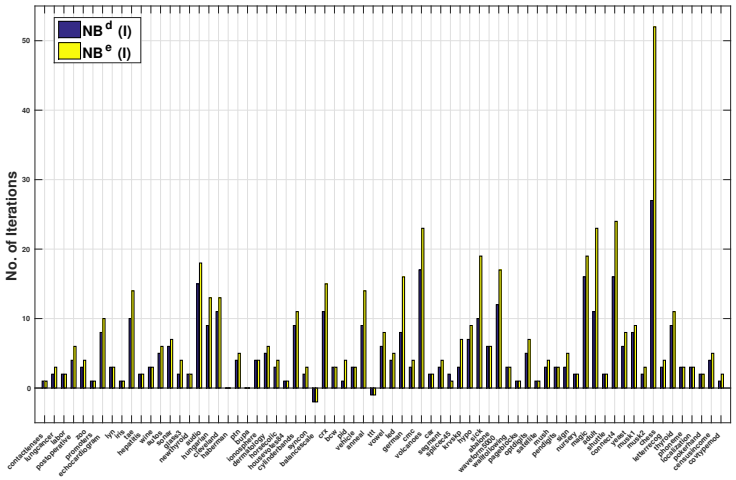

Fig. 9: Number of iterations (after 5 iterations), it takes NB^d and NB^e to reach NLL that NB^w achieved after 5 iterations. If NLL of NB^w is less that that of NB^d or NB^e, number of iterations it takes NB^w to reach that of NB^d (denoted as $NB^{w\text{-}d}$) and that of NB^e (denoted as $NB^{w\text{-}e}$) are plotted with negative sign.

learning with discriminative learning is the goal of future research as will be discussed in Section 8.

8 Conclusion and Future Work

In this paper, we propose an effective parameterization of BN. We present a unified framework for learning the parameters of Bayesian network classifiers. We formulate three different parameterizations and compare their performance in terms of 0-1 Loss, RMSE and training time each parameterization took to converge. We show with NB, TAN and KDB structure that the proposed weighted parameterization has similar 0-1 Loss and RMSE to the other two but significantly faster convergence. We also show that it not only has faster convergence but it also asymptotes to its global minimum much quicker than the other two parameterizations. This is desirable when learning from huge quantities of data with Stochastic Gradient Descent (SGD). It is also shown that discriminative training of BN classifiers also leads to lower bias than the generative parameter learning.

We plan to conduct following future work as the result of this study:

– The three parameterizations presented in this work learn a weight for each attribute-value-per-class-value-per-parent-values. Contrary to $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$, $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ parameterization can generalize parameters. For example, once MAP estimates of probabilities are learned, one can learn a weight: a) for each attribute only (i.e., same weight for all attribute-values, for all

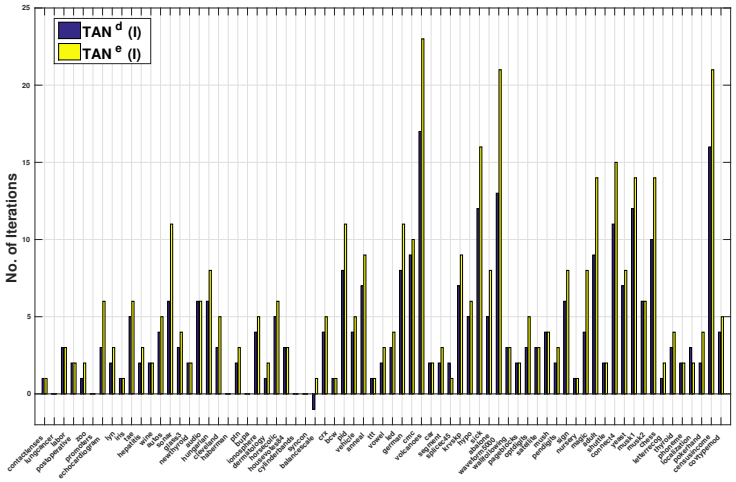

Fig. 10: Number of iterations (after 5 iterations), it takes TAN^{d} and TAN^{e} to reach NLL that TAN^{w} achieved after 5 iterations. If NLL of TAN^{w} is less that that of TAN^{d} or TAN^{e}, number of iterations it takes TAN^{w} to reach that of TAN^{d} (denoted as $\text{TAN}^{\text{w-d}}$) and that of TAN^{e} (denoted as $\text{TAN}^{\text{w-e}}$) are plotted with negative sign.

class values and for all parent values), b) for each attribute-value only, c) for each attribute-value-per-class-value, d) for each attribute-value-per-class-value-per-parent, etc. Such parameter generalization could offer additional speed-up of the training and is a promising avenue for future research.
- How to handle combinations of $\langle$`feature-value`, `parent-value`, `class-value`$\rangle$ that have not been seen at training time is one of the weaker properties of discriminative learning. We plan to design an hierarchical algorithm of discriminative learning that can learn lower-level discriminative weights and can back-off from higher levels if a combination is not observed in the training data.
- We plan to conduct an extended analysis of BN models that can capture higher-order interactions. Because the CLL is not convex for most of these models [19], it falls outside the scope of this paper. This does, however, suggest inviting avenues for big data research, in which context low-bias classifiers are required.

9 Code and Datasets

All the datasets used in this paper are in the public domain and can be downloaded from [5]. Code with running instructions can be download from `https://www.dropbox.com/sh/vd6cma61eibuem0/AACD11ylYnqIXbBWm2DKt7ZYa?dl=0`.

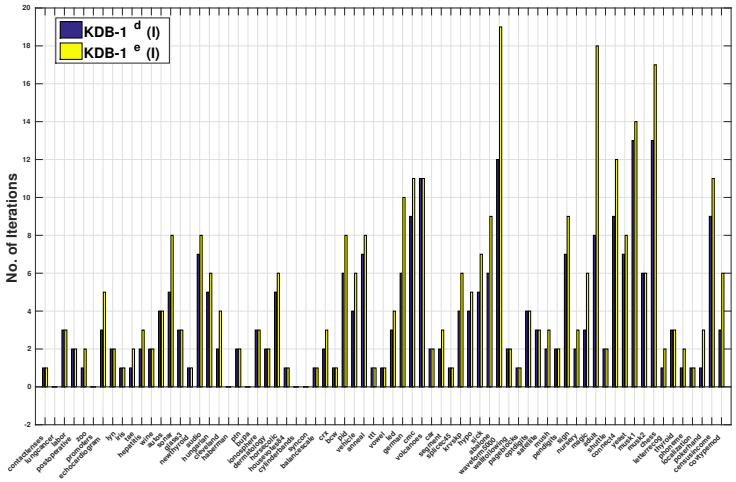

Fig. 11: Number of iterations (after 5 iterations), it takes KDB-1$^{\mathrm{d}}$ and KDB-1$^{\mathrm{e}}$ to reach NLL that KDB-1$^{\mathrm{w}}$ achieved after 5 iterations. If NLL of KDB-1$^{\mathrm{w}}$ is less that that of KDB-1$^{\mathrm{d}}$ or KDB-1$^{\mathrm{e}}$, number of iterations it takes KDB-1$^{\mathrm{w}}$ to reach that of KDB-1$^{\mathrm{d}}$ (denoted as KDB1$^{\mathrm{w\text{-}d}}$) and that of KDB-1$^{\mathrm{e}}$ (denoted as KDB1$^{\mathrm{w\text{-}e}}$) are plotted with negative sign.

10 Acknowledgments

This research has been supported by the Australian Research Council (ARC) under grants DP120100553 and DP140100087, and by the Asian Office of Aerospace Research and Development, Air Force Office of Scientific Research under contracts FA2386-15-1-4007 and FA2386-15-1-4017.

Authors would like to thank Jesus Cerquides, Wray Buntine, Reza Haffari and Ana Martinez for helpful discussion during the course of this paper.

A Proof of Therorem 1

Let us use Lagrange multipliers for constraints in Equation 3 to be placed in Equation 2. Now, we can maximize the resulting objective function: $\mathrm{LL}(\mathcal{B}) + \lambda_0(1 - \sum_{y\in\mathcal{Y}} \theta_{y|\Pi_{0(\mathbf{x})}}) + \sum_i^n \lambda_i(1 - \sum_{x_i\in\mathrm{dom}(X_i)} \theta_{x_i|\Pi_i(\mathbf{x})})$ by first computing its derivative as:

$$\frac{\partial \mathrm{LL}(\mathcal{B})}{\partial \theta_{x_i|\Pi_i(\mathbf{x})}} = \sum_{j=1}^{N} \frac{\mathbf{1}_{x_i^{(j)}=x_i} \mathbf{1}_{y^{(j)}=y} \mathbf{1}_{\Pi_i^{(j)}(\mathbf{x})=\Pi_i(\mathbf{x})}}{\theta_{x_i^{(j)}|\Pi_i(\mathbf{x})}} - \lambda_i.$$

and then setting it to zero. This will lead to $\theta_{x_i|\Pi_i(\mathbf{x})} = \frac{\sum_{j=1}^{N} N_{x_i,y,\Pi_i(\mathbf{x})}}{\lambda_i}$, where $N_{x_i,y,\Pi_i(\mathbf{x})}$ is the empirical count of instances with attribute i taking value x_i, class taking value y and parents taking value $\Pi_i(\mathbf{x})$. Placing $\theta_{x_i|\Pi_i(\mathbf{x})}$ value in Equation 3, we get: $\sum_{x_i\in\mathrm{dom}(X_i)} \frac{\sum_{j=1}^{N} N_{x_i,y,\Pi_i(\mathbf{x})}}{\lambda_i} = 1$, which implies: $\lambda_i = \sum_{x_i\in\mathrm{dom}(X_i)} \sum_{j=1}^{N} N_{x_i,y,\Pi_i(\mathbf{x})}$. Therefore, $\lambda_i = N_{y,\Pi_i(\mathbf{x})}$. Hence we can write:

$$\theta_{x_i|\Pi_i(\mathbf{x})} = \frac{N_{x_i,y,\Pi_i(\mathbf{x})}}{N_{y,\Pi_i(\mathbf{x})}}. \qquad \text{Similarly:} \qquad \theta_{y|\Pi_0(\mathbf{x})} = \frac{N_{y,\Pi_0(\mathbf{x})}}{N_{\Pi_0(\mathbf{x})}}.$$

This equals empirical estimates of probabilities from the data: $\mathrm{P}_{\mathcal{D}}(x_i|\Pi_i(\mathbf{x}))$.

References

1. Buntine, W.: Operations forlearning with graphical models. Journal of Artificial Intelligence Research (2), 159–225 (1994)
2. Byrd, R., Lu, P., Nocedal, J.: A limited memory algorithm for bound constrained optimization. SIAM Journal on Scientific and Statistical Computing **16**(5), 1190–1208 (1995)
3. Carvalho, A., Roos, T., Oliveira, A., Myllymaki, P.: Discriminative learning of Bayesian networks via factorized conditional log-likelihood. Journal of Machine Learning Research (2011)
4. Fayyad, U.M., Irani, K.B.: On the handling of continuous-valued attributes in decision tree generation. Machine Learning **8**(1), 87–102 (1992)
5. Frank, A., Asuncion, A.: UCI machine learning repository (2010). URL `http://archive.ics.uci.edu/ml`
6. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning **29**(2), 131–163 (1997)
7. Greiner, R., Zhou, W.: Structural extension to logistic regression: Discriminative paramter learning of belief net classifiers. In: AAAI (2002)
8. Greiner, R., Zhou, W., Su, X., Shen, B.: Structural extensions to logistic regression: Discriminative parameter learning of belief net classifiers. Journal of Machine Learning Research (2004)
9. Grossman, D., Domingos, P.: Learning bayesian network classifiers by maximizing conditional likelihood. In: ICML (2004)
10. Heckerman, D., Meek, C.: Models and selection criteria for regression and classification. In: International Conference on Uncertainity in Artificial Intelligence (1997)
11. Jebara, T.: Machine Learning: Discriminative and Generative. Springer International Series (2003)
12. Langford, J., Li, L., Strehl, A.: Vowpal wabbit online learning project (2007)
13. Martinez, A., Chen, S., Webb, G.I., Zaidi, N.A.: Scalable learning of bayesian network classifiers. Journal of Machine Learning Research (2015)
14. Ng, A., Jordan, M.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In: Advances in Neural Information Processing Systems (2002)
15. Pernkopf, F., Bilmes, J.: Discriminative versus generative parameter and structure learning of bayesian network classifiers. In: ICML (2005)
16. Pernkopf, F., Bilms, J.A.: Efficient heuristics for discriminative structure learning of bayesian network classifiers. Journal of Machine Learning Research (2010)
17. Pernkopf, F., Wohlmayr, M.: On discriminative parameter learning of bayesian network classifiers. In: ECML PKDD (2009)
18. Ripley, B.D.: Pattern Recognition and Neural Networks. Cambridge University Press (1996)
19. Roos, T., Wettig, H., Grünwald, P., Myllymäki, P., Tirri, H.: On discriminative Bayesian network classifiers and logistic regression. Machine Learning **59**(3), 267–296 (2005)
20. Rubinstein, Y.D., Hastie, T.: Discriminative vs informative learning. In: AAAI (1997)
21. Su, J., Zhang, H., Ling, C., Matwin, S.: Discriminative parameter learning for bayesian networks. In: ICML (2008)
22. Webb, G.I., Boughton, J., Zheng, F., Ting, K.M., Salem, H.: Learning by extrapolation from marginal to full-multivariate probability distributions: decreasingly naive Bayesian classification. Machine Learning pp. 1–40 (2011)
23. Zaidi, N.A., Carman, M.J., Cerquides, J., Webb, G.I.: Naive-bayes inspired effective pre-conditioners for speeding-up logistic regression. In: IEEE International Conference on Data Mining (2014)
24. Zaidi, N.A., Cerquides, J., Carman, M.J., Webb, G.I.: Alleviating naive Bayes attribute independence assumption by attribute weighting. Journal of Machine Learning Research **14**, 1947–1988 (2013)

25. Zaidi, N.A., Webb, G.I., Carman, M.J., Petitjean, F.: Deep broad learning - Big models for Big data. arXiv:1509.01346 (2015)
26. Zhu, C., Byrd, R.H., Nocedal, J.: LBFGSB, fortran routines for large scale bound constrained optimization. ACM Transactions on Mathematical Software **23**(4), 550–560 (1997)