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.

In this paper, we show that:

- The proposed formalization of the parameter learning task for BN is actually a re-parameterization of the one step (discriminative) learning problem (this will become clear when we introduce the proposed framework) 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. In Section 5.5, we will discuss our proposed approach from the perspective of pre-conditioning in unconstrained optimization problems.
- The proposed approach results in a three-level hierarchy of nested parameterizations, where each additional level introduces (or "unties") exponentially more parameters in order to fit ever smaller violations of independence.
- Regularization of the discriminative parameters in the proposed discriminative learning approach allows to limit the amount of allowable violation of independence and effectively interpolate between discriminative and generative parameter estimation.

The rest of this paper is organized as follows. In Section 2, we present our proposed framework for parameter learning of Bayesian network classifiers. We also give the formulation for class-conditional Bayesian Network models (CCBN) in this section. Two established 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.

All the symbols used in this work are listed in Table 1.

2 A Simple Framework for Parameter Learning of BN 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 a variable, Z_i), and a set of parameters Θ, that quantifies the dependencies within the structure. The variables are partitioned into a single target, the class variable $Y{=}Z_0$ and n covariates $X_1{=}Z_1, X_2{=}Z_2, \ldots X_n{=}Z_n$, called the *attributes*. The parameter Θ, contains

Notation	Description
n	Number of attributes
N	Number of data points in $\mathcal{D}$
$\mathrm{P}(e)$	Probability of event e
$\mathrm{P}(e \mid g)$	Conditional probability of event e given g
$\hat{\mathrm{P}}(.)$	An estimate of $\mathrm{P}(.)$
$\mathcal{D} = \{\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(N)}\}$	Data consisting of N objects
$\mathcal{L} = \{y^{(1)}, \ldots, y^{(N)}\}$	Labels of data points in $\mathcal{D}$
$\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$	An object (n-dimensional vector of attribute values)
$\mathbf{z} = \langle z_0, z1, \ldots z_n \rangle$	A labeled object, where $z_0 = y^{(i)}, z_1 = x_1, \ldots, z_n = x_n$ and $\mathbf{x}^{(i)} \in \mathcal{D}$
Y	Random variable associated with class label
y	$y \in Y$. Class label for object. Same as z_0
$\lvert Y \rvert$	Number of classes
X_i	Random variable associated with attribute i
x_i	$x_i \in X_i$. i-th attribute value
$\lvert X_i \rvert$	Number of values of attribute X_i
Z_i	Random variable associated with attribute i, or in the case of Z_0, the class.
z_i	$z_i \in Z_i$. i-th attribute value, or for z_0, the class
$\lvert Z_i \rvert$	Number of values of variable Z_i
$\mathcal{B}$	Bayesian Network (directed acyclic graph), parameterized by Θ
$\mathcal{B}^*$	Class-conditional BN based on $\mathcal{B}$, parameterized by $\boldsymbol{\theta}$
$\mathcal{G}$	Structure of BN $\mathcal{B}$
Θ	Set of parameters associated with $\mathcal{B}$
$\mathrm{P}_{\mathcal{B}}(.)$	Probability is based on BN $\mathcal{B}$
$\Pi_i(.)$	Function taking $\mathbf{z}$ as an input, returns the values of the attributes which are the parents of i
$\Pi_0(.)$	Parents of class
$\theta_{Z_i=z_i \mid \Pi_i(\mathbf{z})}$	Probability of $Z_i = z_i$ given its parents
$\theta_{z_i \mid \Pi_i(\mathbf{z})}$	Short form of $\theta_{Z_i=z_i \mid \Pi_i(\mathbf{x})}$
$\theta_{z_i:j \mid y:k, \Pi_i:l}$	Probability of variable i taking value j, class (y) taking value k and its parents (Π_i) taking value l
$\beta_{y,x_i,\Pi_i(\mathbf{x})}$	Parameter associated with class y, attribute i taking value x_i and i's parent's-values Π_i
β_{y,x_i,Π_i}	Same as $\beta_{y,x_i,\Pi_i(\mathbf{x})}$
$\beta_{x_i:j,y:k,\Pi_i:l}$	Parameter associated with attribute i taking value j, class (y) taking value k and its parents (Π_i) taking value l
$\boldsymbol{\theta}, \mathbf{w}, \beta$	Vector of θ, w and β parameters respectively
$N_{x_i,y,\Pi_i(\mathbf{x})}$	Empirical count of data with attribute i taking value x_i, class taking value y and parents taking value $\Pi_i(\mathbf{x})$

Table 1: List of symbols used.

a set of parameters for each vertex in $\mathcal{G}$: $\theta_{z_0 \mid \Pi_0(\mathbf{x})}$ and for $1 \leq i \leq n$, $\theta_{z_i \mid y, \Pi_i(\mathbf{x})}$, where $\Pi_i(.)$ is a function which given the datum $\mathbf{x} = \langle x_1, x_1, \ldots, x_n \rangle$ as its input, returns the values of the attributes that are the parents of node i in structure $\mathcal{G}$. For notational simplicity, instead of writing $\theta_{Z_0=z_0 \mid \Pi_0(\mathbf{x})}$ and $\theta_{Z_i=z_i \mid y, \Pi_i(\mathbf{x})}$, we write $\theta_{z_0 \mid \Pi_0(\mathbf{x})}$ and $\theta_{z_i \mid y, \Pi_i(\mathbf{x})}$. A BN $\mathcal{B}$ computes the joint probability distribution as: $\mathrm{P}_{\mathcal{B}}(y, \mathbf{x}) = \theta_{z_0 \mid \Pi_0(\mathbf{x})} \cdot \prod_{i=1}^{n} \theta_{z_i \mid y, \Pi_i(\mathbf{x})}$.

For a BN $\mathcal{B}$, we can write:

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

Now, the corresponding conditional distribution $\mathrm{P}_{\mathcal{B}}(y|\mathbf{x})$ can be computed with the Bayes rule as:

$$\begin{aligned} \mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) &= \frac{\mathrm{P}_{\mathcal{B}}(y, \mathbf{x})}{\mathrm{P}_{\mathcal{B}}(\mathbf{x})}, \\ &= \frac{\theta_{y|\Pi_0(\mathbf{x})} \prod_{i=1}^{n} \theta_{x_i|y,\Pi_i(\mathbf{x})}}{\sum_{y' \in \mathcal{Y}} \theta_{y'|\Pi_0(\mathbf{x})} \prod_{i=1}^{n} \theta_{x_i|y',\Pi_i(\mathbf{x})}}. \end{aligned} \tag{2}$$

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:

$$\begin{aligned} \mathrm{LL}(\mathcal{B}) &= \sum_{j=1}^{N} \log \mathrm{P}_{\mathcal{B}}(y^{(j)}, \mathbf{x}^{(j)}), \\ &= \sum_{j=1}^{N} \left(\log \theta_{y^{(j)}|\Pi_0(\mathbf{x}^{(j)})} + \sum_{i=1}^{n} \log \theta_{x_i^{(j)}|\Pi_i(\mathbf{x}^{(j)})} \right), \end{aligned} \tag{3}$$

$$\text{with} \quad \sum_{y \in \mathcal{Y}} \theta_{y|\Pi_0(\mathbf{x})} = 1, \quad \text{and} \quad \sum_{x_i \in \mathcal{X}_i} \theta_{x_i|\Pi_i(\mathbf{x})} = 1. \tag{4}$$

Maximizing Equation 3 to optimize the parameters (θ) is the maximum-likelihood estimation of the parameters.

Theorem 1 *Within the constraints in Equation 4, Equation 3 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 3 (and fulfilling the constraints in Equation 4) are typically known as 'Generative' estimates of the probabilities.

2.2 Class-Conditional BN (CCBN) Models

Instead of following a two-step process for classification with BN, where step 1 involves maximizing $\mathrm{P}(y, \mathbf{x})$ and the second step is application of Bayes rule to obtain $\mathrm{P}(y|\mathbf{x})$, one can directly optimize for $\mathrm{P}(y|\mathbf{x})$ by maximizing the Conditional Log-Likelihood (CLL). Optimizing CLL is generally considered a

more effective objective function (for classification) 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:

$$\begin{aligned} &= \sum_{j=1}^{N} \left(\log \mathrm{P}_{\mathcal{B}}(y^{(j)}, \mathbf{x}^{(j)}) - \log \sum_{y'}^{|\mathcal{Y}|} \mathrm{P}_{\mathcal{B}}(y', \mathbf{x}^{(j)}) \right) \\ &= \sum_{j=1}^{N} \left(\log \theta_{y^{(j)}|\Pi_0(\mathbf{x}^{(j)})} + \sum_{i=1}^{n} \log \theta_{x_i^{(j)}|\Pi_i(\mathbf{x}^{(j)})} \right) - \\ &\qquad \log \left(\sum_{y'}^{|\mathcal{Y}|} \theta_{y'|\Pi_0(\mathbf{x}^{(j)})} \prod_{i=1}^{n} \theta_{x_i|y',\Pi_i(\mathbf{x}^{(j)})} \right). \end{aligned} \tag{5}$$

The only difference between Equation 3 and Equation 5 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 5 are not the same as those that maximize Equation 3. We provide two intuitions as to why maximizing the CLL should 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 [23]. 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 [22]:

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 2.

This means that the nodes in $\mathcal{B}^*$ are nodes comprising only the Markov blanket of the class y. However, for most BN classifiers the class has no parents and is made a parent of all attributes. This has the effect that every attribute is in the Markov blanket of the class.

We will assume that the parents of the 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.

2.3 A Simple Framework

It is no exaggeration to say that Equation 1 has a pivotal role in BN classification. Let us modify Equation 1 by introducing an extra set of parameter, say w for every parameter θ. Let $\boldsymbol{\theta}$ and $\mathbf{w}$, represent the vectors of all θ and w parameters. In the following, let us also make a distinction between 'Fixed' and 'Optimized' parameters – during the optimization process. Parameters that are optimized are referred to as *Optimized* parameters and parameters that do not change their value during the optimization process are referred to as *Fixed*. Now, we can write:

$$\mathrm{P}_{\mathcal{B}}(y, \mathbf{x}) = \theta_y^{w_y} \prod_{i=1}^{n} \theta_{x_i|y,\Pi_i(\mathbf{x})}^{w_{x_i,y,\Pi_i(\mathbf{x})}}. \tag{6}$$

Optimizing to compute class-probabilities using Equation 6, there are several possibilities, of which we will discuss only four in the following:

1. **Generative**: Initialize $\mathbf{w}$ with 1 and treat it as fixed parameter. Treat, $\boldsymbol{\theta}$ as optimized parameter and optimize it with the generative objective function as given in Equation 3.
2. **Discriminative**: Initialize $\mathbf{w}$ with 1 and treat it as fixed parameter. Treat, $\boldsymbol{\theta}$ as an optimized parameter and optimize it with the discriminative objective function as given in Equation 5. As discussed, this results in adding a normalization term to convert $\mathrm{P}(y, \mathbf{x})$ in Equation 6 to $\mathrm{P}(y|\mathbf{x})$. We denote this 'discriminative CCBN' and describe it in detail in Section 3.
3. **Discriminative**: Initialize $\mathbf{w}$ with 1 and treat it as a fixed parameter. Treat, $\boldsymbol{\theta}$ as an optimized parameter and optimize it with the discriminative objective function as given in Equation 5, but constrain parameter $\boldsymbol{\theta}$ to be actual probabilities. We denote this 'extended CCBN' and provide a detailed description in Section 4.
4. **Discriminative**: Two step learning. In the first step, initialize $\mathbf{w}$ with 1 and treat it as a fixed parameter. Treat, $\boldsymbol{\theta}$ as an optimized parameter and optimize it with the generative objective function as given in Equation 3. In the second step, treat $\boldsymbol{\theta}$ as a fixed parameter and optimize for $\mathbf{w}$ using a discriminative objective function. This approach is inspired from the fact that weights $\mathbf{w}$ in Equation 6 are set through generative learning, unlike discriminative and extended CCBN, where it is set to one. We denote this 'weighted CCBN' and describe it in detail in Section 5.

A brief summary of these parameterizations is also given in Table 2.

	Generative – Maximize LL	**Discriminative** – Maximize CLL	**Discriminative** – Maximize CLL	**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	$\theta_y \prod_{i=1}^{n} \theta_{x_i \mid y, \Pi_i(\mathbf{x})}$	$\frac{\theta_y \prod_{i=1}^{n} \theta_{x_i \mid y, \Pi_i(\mathbf{x})}}{\sum_{y'}^{\mid \mathcal{Y} \mid} \theta_{y'} \prod_{i=1}^{n} \theta_{x_i \mid y', \Pi_i(\mathbf{x})}}$	$\frac{\exp(\beta_y + \sum_{i=1}^{n} \beta_{y, x_i, \Pi_i})}{\sum_{y'=1}^{\mid \mathcal{Y} \mid} \exp(\sum_{y'} \beta_{y'} + \sum_{i=1}^{n} \beta_{y', x_i, \Pi_i})}$	$\frac{\theta_y^{w_y} \prod_{i=1}^{n} \theta_{x_i \mid y, \Pi_i(\mathbf{x})}^{w_{y, x_i, \Pi_i}}}{\sum_{y'}^{\mid \mathcal{Y} \mid} \theta_{y'}^{w_{y'}} \prod_{i=1}^{n} \theta_{x_i \mid y', \Pi_i(\mathbf{x})}^{w_{y', x_i, \Pi_i}}}$
Pre-step	None	None	None	Optimize for $\boldsymbol{\theta}$ using Generative objective function of Equation 3
Constraints	$\theta_y \in [0,1]^{\mid \mathcal{Y} \mid}$, $\theta_{i,y} \in [0,1]^{\mid \mathcal{X}_i \mid}$	$\theta_y \in [0,1]^{\mid \mathcal{Y} \mid}$, $\theta_{i,y} \in [0,1]^{\mid \mathcal{X}_i \mid}$	None	$\theta_y \in [0,1]^{\mid \mathcal{Y} \mid}$, $\theta_{i,y} \in [0,1]^{\mid \mathcal{X}_i \mid}$
Optimized Param.	$\boldsymbol{\theta}$	$\boldsymbol{\theta}$	$\boldsymbol{\beta}$	$\mathbf{w}$
'Fixed' Param.	None	None	None	$\boldsymbol{\theta}$

Table 2: Comparison of different parameter learning techniques for Bayesian Network Classifiers.

3 Parameterization 1: Discriminative CCBN Model

Logistic regression (LR) is the CCBN model associated to the NB structure optimizing Equation 2. 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 [15,31]. Inspired from [22], 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 2 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 5 and write it on a per datum basis as:

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

In light of Definition 2, let us define a parameter $\beta_\bullet$ that is associated with each parameter $\theta_\bullet$ in Equation 7, 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 7 can be written as:

$$\mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = \frac{\exp(\beta_y + \sum_{i=1}^{n} \beta_{y,x_i,\Pi_i})}{\sum_{y'=1}^{|\mathcal{Y}|} \exp(\sum_{y'} \beta_{y'} + \sum_{i=1}^{n} \beta_{y',x_i,\Pi_i})}. \tag{8}$$

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:

$$\log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = (\beta_y + \sum_{i=1}^{n} \beta_{y,x_i,\Pi_i}) - \log(\sum_{y'=1}^{|\mathcal{Y}|} \exp(\beta_{y'} + \sum_{i=1}^{n} \beta_{y',x_i,\Pi_i})). \tag{9}$$

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 9 can be computed as:

$$\frac{\partial \log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x})}{\partial \beta_{y:k}} = (\mathbf{1}_{y=k} - \mathrm{P}(k|\mathbf{x})), \tag{10}$$

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

$$\frac{\partial \log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x})}{\partial \beta_{y:k,x_i:j,\Pi_i:l}} = (\mathbf{1}_{y=k} - \mathrm{P}(k|\mathbf{x}))\, \mathbf{1}_{x_i=j} \mathbf{1}_{\Pi_i=l}, \tag{11}$$

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 [10], 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 [10] – 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 4 and is obtained by maximizing the CLL in Equation 5.

Let us re-define $\mathrm{P}_{\mathcal{B}}(y|\mathbf{x})$ in Equation 5 on a per-datum-basis as:

$$\log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = (\log \theta_y + \sum_{i=1}^{n} \log \theta_{x_i|y,\Pi_i(\mathbf{x})}) - \log \sum_{y'}^{|\mathcal{Y}|} (\theta_{y'} \prod_{i=1}^{n} \theta_{x_i|y',\Pi_i(\mathbf{x})}). \tag{12}$$

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:

$$\theta_{x_i:j'|y:k,\Pi_i:l} = \frac{\exp(\beta_{x_i:j'|y:k,\Pi_i:l})}{\sum_{j''} \exp(\beta_{x_i:j''|y:k,\Pi_i:l})}. \tag{13}$$

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:

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

Applying the chain rule:

$$\begin{aligned}\frac{\partial\log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x})}{\partial\beta_{x_i:j|y:k,\Pi_i:l}} &= \sum_{j'}\frac{\partial\log \mathrm{P}(y|\mathbf{x})}{\partial\theta_{x_i:j'|y:k,\Pi_i:l}}\frac{\partial\theta_{x_i:j'|y:k,\Pi_i:l}}{\partial\beta_{x_i:j|y:k,\Pi_i:l}},\\ &= (\mathbf{1}_{y=k}\mathbf{1}_{x_i=j}\mathbf{1}_{\Pi_i=l}-\mathbf{1}_{x_i=j}\mathbf{1}_{\Pi_i=l}\mathrm{P}(k|\mathbf{x})) -\\ \theta_{x_i:j|y:k,\Pi_i:l}&\sum_{j'}\left(\mathbf{1}_{y=k}\mathbf{1}_{x_i=j'}\mathbf{1}_{\Pi_i=l}-\mathbf{1}_{x_i=j'}\mathbf{1}_{\Pi_i=l}\mathrm{P}(k|\mathbf{x})\right),\end{aligned} \tag{14}$$

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 13 to obtain the desired parameters of extended CCBN. Note that Equation 14 corresponds to Equation 11. The only difference is the presence of the normalization term that is subtracted from the gradient in Equation 14.

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

Inspired from [28], 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 2 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 2 to incorporate weights as:

$$\mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = \frac{\theta_y^{w_y}\prod_{i=1}^n\theta_{x_i|y,\Pi_i(\mathbf{x})}^{w_{y,x_i,\Pi_i}}}{\sum_{y'}^{|\mathcal{Y}|}\theta_{y'}^{w_{y'}}\prod_{i=1}^n\theta_{x_i|y',\Pi_i(\mathbf{x})}^{w_{y',x_i,\Pi_i}}}. \tag{15}$$

The corresponding weighted CLL can be written as:

$$\log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = (w_y \log \theta_y + \sum_{i=1}^{n} w_{y,x_i,\Pi_i} \log \theta_{x_i|y,\Pi_i(\mathbf{x})}) - \log \sum_{y'}^{|\mathcal{Y}|} (\theta_{y'}^{w_y} \prod_{i=1}^{n} \theta_{x_i|y',\Pi_i(\mathbf{x})}^{w_{y,x_i,\Pi_i}}). \tag{16}$$

Note, that Equation 16 is similar to Equation 12 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 16 can be computed as:

$$\frac{\partial \log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x})}{\partial w_{y:k}} = (\mathbf{1}_{y=k} - \mathrm{P}(k|\mathbf{x})) \log \theta_{y|\Pi_0(\mathbf{x})}, \tag{17}$$

for the class y, while for the other parameters:

$$\frac{\partial \log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x})}{\partial w_{y:k,x_i:j,\Pi_i:l}} = (\mathbf{1}_{y=k} - \mathrm{P}(k|\mathbf{x}))\, \mathbf{1}_{x_i=j} \mathbf{1}_{\Pi_i=l} \log \theta_{x_i|y,\Pi_i(\mathbf{x})}. \tag{18}$$

One can see that Equations 17 and 18 correspond to Equations 10 and 11. The only difference between them is the presence of the $\log \theta_{\bullet}$ factor in the $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ case.

5.1 On Initialization of Parameters

Initialization of the parameters, which sets the starting point for the optimization, is critical to the speed of convergence and will be addressed in this section. Obviously, a better starting point (in terms of CLL), will make the optimization easier and conversely, a worse starting point will make optimization harder. In this paper, we will study two different starting points for the parameters:

Initialization with Zeros This is the standard initialization where all the optimized parameters are initialized with 0 [21].

Initialization with 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 initializing the $\boldsymbol{\theta}$ parameter with the generative estimates for Parameterizations 1 and 2 ($\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$), and initializing the $\mathbf{w}$ parameter to 1 for Parameterization 3 ($\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$).

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

5.2 Comment on Regularization

$\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ parameterization offers an elegant framework for blending discriminatively and generatively learned parameters. With regularization, one can indeed 'interpolate' between the two sets of parameters. Traditionally, one regularizes parameters towards 0 to prevent over-fitting. For example, let us modify Equation 15 to integrate an L2-regularization:

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

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 1 rather than zero. In this case, we can write:

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

Doing so allows the regularization parameter λ to be used to 'pull' the dicriminative estimates toward the generative ones. A very small value of λ results in optimized parameter $\mathbf{w}$ dominating the determination of $\mathrm{P}(y|\mathbf{x})$, whereas, a very large value of λ pulls $\mathbf{w}$ towards 1 and, therefore, the fixed parameters will dominate the class-conditional probabilities. Regularization for $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ remains an area for future research, but we conjecture that one can tune a value of λ (for example through cross-validation) to attain better performance than can be achieved by either generative or discriminative parameters alone. Once could also interpret the regularization parameter as controlling the amount of *independence violation* between the discriminative and generative models.

5.3 Optimizing Discriminative/Generative Parameterization

There are great advantages in optimizing an objective function that is convex. The convexity of the three discriminative parameterizations that we have discussed depends on the underlying structure of the CCBN ($\mathcal{M}^{\mathcal{B}^*}$). From [22], it follows that optimizing a CCBN parameterized by either $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$, $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ or $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ leads to a convex optimization problem if and only if the structure has no immoral nodes. In other words, the optimization problem is convex if and only if parents of all the nodes are are connected with each other. This constraint is true for a number of popular BN classifiers including NB, TAN and KDB ($K = 1$), but not true for general BN or for KDB structures with $K > 1$. Therefore, in this work, we have used only limited BN structures such as NB, TAN and KDB ($K = 1$). Investigation of the application of our approach to more complex moral structures is a promising topic for future work. We note

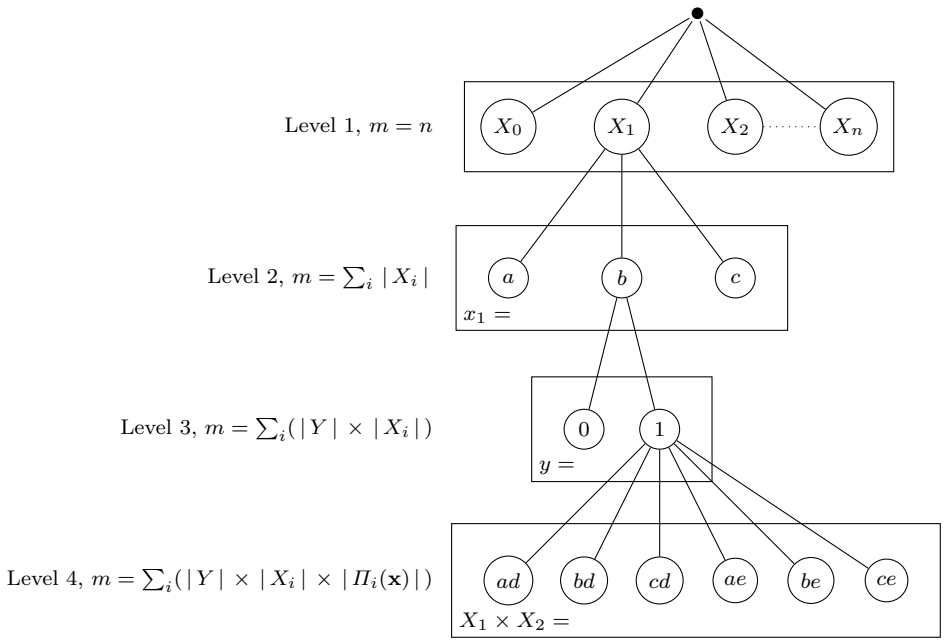


Fig. 1: Depiction of various levels in parameter nesting, along with number of parameters (m) to be optimized at each level. Note that only one node per level is expanded, for illustration. Attribute X_1 takes values $\{a, b, c\}$ and takes class $Y = \{0, 1\}$ and attribute $X_2 = \{d, e, f\}$ as parents.

in passing, that a similar two step discriminative parameterization has also been shown to be effective for the non-convex objective function mean-square-error [30].

5.4 Nested Parameterizations

One can see that learning $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$, $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ models can lead to a large number of parameters that need to be optimized, even on moderate size datasets. One can, however, nest these parameters. The idea is to exploit relationships between parameters so that the number of parameters that need to be optimized are reduced significantly. Figure 1 depicts four levels of parameter nesting. The first level entails learning a parameter for each attribute. The second level entails learning a parameter for every attribute-value. The next level learns a parameter for every attribute-value-per-class-value. The final level (Level 4) is the most comprehensive case. It entails learning a parameter for every attribute-value-per-class-per-parent-value.

Nesting as shown in Figure 1, though effective, is not very intuitive for $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$. For example, doing a logistic regression by learning a parameter associated only with the attributes will result in optimizing fewer parameters

but might not be effective in terms of classification accuracy. However, the hierarchy of models applies naturally to $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$. $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ incorporates learning initial parameters by optimizing the LL objective function. Therefore, the searched parameters optimized in the second step can be nested effectively. For example, Level 1 weighting in Figure 1 can be seen as alleviating the conditional attribute independence assumption (CAIA) between attributes. Similarly, Level 2 will have the effect of binarizing each attribute, and alleviating CAIA between new attributes. In the following we will derive the respective gradients for each level from the most comprehensive case of Level 4.

Gradients for Level 4 are given in Equation 17 and 18. Level 3 corresponds to learning a weight-per-attribute-value-per-class. The weight vector in this case will be of the size $m = \sum_i(\,|\,Y\,|\,\times\,|\,X_i\,|\,)$. The gradients with respect to new weight vectors can be obtained in the following way:

$$\frac{\partial \log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x})}{\partial w_{x_i:j|y:k}} = (\mathbf{1}_{y=k} - \mathrm{P}(k|\mathbf{x}))\mathbf{1}_{x_i=j} \log \theta_{x_i|y,\Pi_i(\mathbf{x})}. \tag{19}$$

Level 2 weighting corresponds to learning a weight-per-attribute-value. We can compute the gradient with respect to the weight vector of size $m = \sum_i \,|\,X_i\,|$, as:

$$\frac{\partial \log \mathrm{P}_{\mathcal{B}}(y|\mathbf{x})}{\partial w_{x_i:j}} = (\log \theta_{x_i|y,\Pi_i(\mathbf{x})} - \sum_{y'} \mathrm{P}(y'|\mathbf{x}) \log \theta_{x_i|y',\Pi_i(\mathbf{x})})\mathbf{1}_{x_i=j}. \tag{20}$$

Similarly, learning a weight-per-attribute leads to a weight vector of size m and can its gradients can be obtained as:

$$\frac{\partial \log \mathrm{P}\mathcal{B}(y|\mathbf{x})}{\partial w_i} = \log \theta_{x_i|y,\Pi_i(\mathbf{x})} - \sum_{y'} \mathrm{P}(y'|\mathbf{x}) \log \theta_{x_i|y',\Pi_i(\mathbf{x})}. \tag{21}$$

Now, one can control the bias and variance of the classifier by selecting between three different levels of parameterization with ever greater model complexity.

5.5 Discussion

5.5.1 Pre-conditioning - Why is our technique helpful?

It can be seen that $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ results in re-scaling of $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ parameterization. What is the effect of this re-scaling on the model? Since there is no closed-form solution, we optimize the CLL with first-order gradient-descent methods, such as gradient descent, conjugate gradient, quasi-Newton (L-BGFS) or Stochastic Gradient Descent. These are all affected by scaling[1]. We use the generative estimates as an effective pre-conditioning method.

[1] Note that second-order algorithms such as the Newton method are not affected by scaling, but they are often computationally impractical because they require computation and inversion of the Hessian at each step.

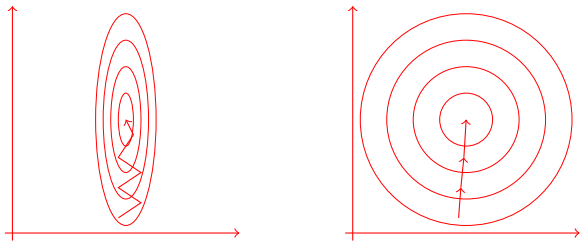


Fig. 2: On the importance of scaling for first-order gradient descent methods. Axes represent two possible β_1 and β_2. Left: non-scaled space. Right:Re-scaled.

A pre-conditioner converts an ill-conditioned problem into a better conditioned one, such that the gradient of the objective function is uniform across all dimensions. A better conditioned optimization problem has a better convergence profile. This is because if different parameters have significantly different "influence" on the objective function, then the gradient does not point directly towards the minimum that is the objective of the optimization process. We illustrate this in Figure 2 where we show the contour plot of the CLL for different β. We can see that when the CLL has an 'elliptical' shape with respect to the parameters, then the gradient is not oriented directly towards the objective and each step makes only partial progress in the true direction of the final objective. Our re-scaling improves the orientation of the gradient speeding convergence.

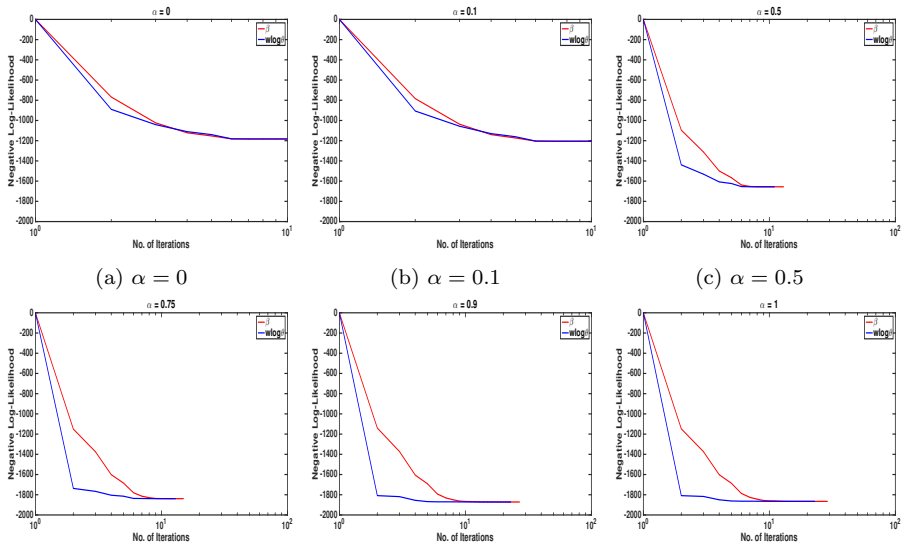
Note that it is the relative scaling of the axes that affects the orientation of the gradient. Isotropic scaling (that is, scaling all axes uniformly) has no effect on convergence.

To further demonstrate our point, we perform a simple experiment with synthetic data that we generate so that the CLL is more or less "elliptical". We use with three binary features and two class values. We sample the covariates randomly and uniformly and use a simple logistic regression model, which corresponds to our framework using Naive Bayes as the BN structure. The class distribution is given by

$$\mathrm{P}(y \,|\, x_1, x_2, x_3) = \frac{1}{\exp(-(10^{\alpha} \cdot x_1 + 1 \cdot x_2 + 10^{-\alpha} \cdot x_3))}.$$

By increasing the value of α, we increase the elongation of the CLL space. When $\alpha = 0$, the three features contribute uniformly to the class prediction and it is a well-conditioned problem. We can then expect pre-conditioning to have little to no influence on the convergence. As $\alpha \to 1$, the problem becomes very ill-conditioned. On such problems, pre-conditioning will have greatest effect.

We compare the convergence profile of vanilla LR (discriminative CCBN) and our pre-conditioned weighted CCBN with Naive Bayes structure (which is associated to a LR model) by varying α from 0 to 1. For each dataset



(a) $\alpha = 0$ (b) $\alpha = 0.1$ (c) $\alpha = 0.5$

(d) $\alpha = 0.75$ (e) $\alpha = 0.9$ (f) $\alpha = 1$

Fig. 3: Comparison of rate of convergence on the three synthetic datasets by varying α. The X-axis is on log scale. Parameters are initialized to zero.

10,000 data points were generated. We report the convergence results in Figure 3. These confirm the explanation given above. The benefit of our technique progressively increases as the relative influence of the covariates on the class increases. We will show in Section 7 that this is the case for the vast majority of real-world datasets.

5.5.2 Is this an over-parametrised model?

It can be seen that the $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ parameterization is based on Equation 6 – which is over-parameterized in the sense that there are twice as many parameters specifying the likelihood as would strictly be necessary. The question is: do we benefit from having both $\mathbf{w}$ and $\boldsymbol{\theta}$ parameters? In this section, we will discuss the implications of introducing $\mathbf{w}$ parameters to the following vanilla (per-datum) likelihood (on which $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ is based on):

$$\mathrm{P}_{\mathcal{B}}(y, \mathbf{x}) = \theta_y \prod_{i=1}^{n} \theta_{x_i|y,\Pi_i(\mathbf{x})}, \tag{22}$$

If one goal of weighted CCBN is to combine generative and discriminative learning by using over-parameterized likelihood in Equation 6, one could do the following two-step learning. In step 1, one can learn the $\boldsymbol{\theta}$ by optimizing a generative objective function, and in the second step, optimize a discriminative objective function but initialize the $\boldsymbol{\theta}$ parameters with the parameters that were obtained in step 1. In fact, this should be a recommended procedure

to speed-up discriminative training (for discriminative CCBN and extended CCBN) as it is often effective in practice.However, one should notice that in this case, the discriminative learning model does start from the estimates of parameters that were obtained from generative learning, but once an iterative step is taken for discriminative learning, the generative estimates are lost, and have no further influence on the discriminative learning process.

6 Related Work

There have been several comparative studies of discriminative and generative structure and parameter learning of Bayesian Networks [9,11,18]. 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 [9]. 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 that leads to faster discriminative learning.

An approach for discriminative learning of the parameters of BN based on discriminative computation of pseudo-frequencies from the data is presented in [25]. *Discriminative Frequency Estimates* (DFE) are computed by injecting a discriminative element to generative computation of the probabilities. During the pseudo-frequencies computation process, rather than using empirical frequencies, DFE estimates how well the current classifier does on each data point and then updates the frequency 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 [7]. The experiments are conducted on the datasets described in Table 3.

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

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 3: Details of Datasets (UCI Domains)

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. The reason for performing bias/variance estimation is to investigate if optimizing a discriminative function leads to a lower bias classifier or not. There are a number of different bias-variance decomposition definitions. In this research, we use the bias and variance definitions of [14] together with the repeated cross-validation bias-variance estimation method proposed by [26]. Kohavi and Wolpert [14] define bias and variance as follows:

$$\text{bias}^2 = \frac{1}{2} \sum_{y \in \mathcal{Y}} \left(\mathrm{P}(y|\mathbf{x}) - \hat{\mathrm{P}}(y \,|\, \mathbf{x}) \right)^2 ,$$

and

$$\text{variance} = \frac{1}{2} \left(1 - \sum_{y \in \mathcal{Y}} \hat{\mathrm{P}}(y \,|\, \mathbf{x})^2 \right) .$$

The reason for reporting 0-1 Loss and RMSE is to investigate if the proposed parameterization $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ leads to a comparable performance to $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ parameterizations and also to determine how much performance gain is achieved over generative learning. We will also evaluate parameterizations in terms of training time (measured in seconds) and number of iterations it takes each parameterization to converge.

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 $p \leq 0.05$. Significant results are shown in bold font in the table.

We report results on two categories of datasets. The first category, labeled *All*, consists of all datasets in Table 3. The second category, labeled *Big*, consists of datasets that have more than 10000 instances. The reason for splitting datasets into two categories is to show explicitly the effectiveness of our proposed optimization on bigger datasets[2]. It is only on big datasets, that each iteration is expensive and, therefore, any technique that leads to faster and better convergence is highly desirable. Note, that we do not expect the three discriminative parameterizations $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$, $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ to differ in their prediction accuracy. That is, we should expect a similar spread of both 0-1 Loss and RMSE values. However, we should be interested in each parameterization's convergence profile and the training time.

Numeric attributes are discretized using the Minimum Description Length (MDL) discretization method [6]. 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] using the original implementation available at `http://users.eecs.northwestern.edu/~nocedal/lbfgsb.html`. Following standard procedures [32], 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 number of iterations exceeds 10^4.

We experiment with three Bayesian network structures that is: naive Bayes (NB), Tree-Augmented naive Bayes (TAN) [8] and k-Dependence Bayesian Network (KDB) with $K = 1$ [24]. Naive Bayes, is a well-known classifier which is based on the assumption that when conditioned on the class, attributes are independent. Tree-Augmented Naive Bayes augments the NB structure by allowing each attribute to depend on at most one non-class attribute. It relies on an extension of the Chow-Liu tree [4], that utilizes conditional mutual information (between pairs of attributes given the class) to find a maximum spanning tree over the attributes in order to determine the parent of each. Similarly, in KDB, each attribute takes k attributes plus the class as its parents. The attributes are selected based on their mutual information with the class. Then, the parent of an attribute i is chosen that maximizes the conditional mutual information of attribute i and parent j given the class that is: $\mathrm{argmax}_j \mathrm{CMI}(X_i, X_j|Y)$.

[2] By big, we mean datasets that have large number of instances, rather than large number of features.

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 5.1, we initialize the parameters to the log of the MAP estimates (or parameters optimized by generative learning). The following naming convention is used in the results:

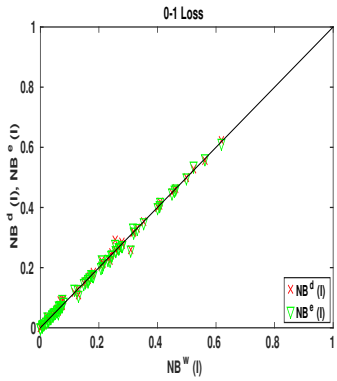
- The '(I)' in the label represents this initialization
- An absence of '(I)' means the parameters are initialized to zero.

7.1 NB Structure

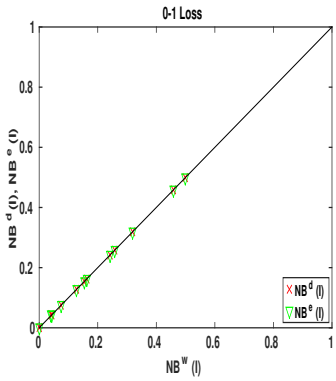
Comparative scatter plots on all 72 datasets for 0-1 Loss, RMSE and training time values for NB^{w}, NB^{d} and NB^{e} are shown in Figure 4. Training time plots are on the log scale. The plots are shown separately for *Big* datasets. It can be seen that the three parameterizations have a similar spread of 0-1 Loss and RMSE values, however, NB^{w} is greatly advantaged in terms of its training time. We will see in Section 7.4 that this computational advantage arises due to the desirable convergence property of NB^{w}. 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}. Slight variation in the accuracy of three discriminative parameterizations (that is 0-1 Loss and RMSE performance) on *All* datasets is due to the numerical instability of the solver use for optimization. The difference mainly arises on very small datasets. It can be seen that on *big* datasets, the three parameterizations result in the same accuracy.

The geometric means of the 0-1 Loss and RMSE results are shown in Figure 5. It can be seen that the three discriminative parameterizations, especially on *Big* datasets, has much better performance (both 0-1 Loss and RMSE) than the generative learning.

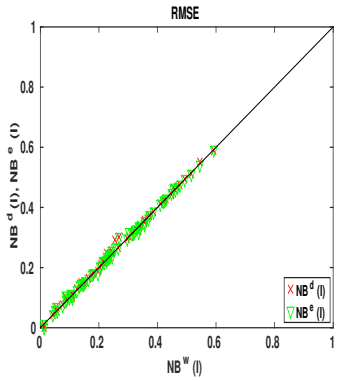
The training time comparison is given in Figure 6a. Note that the training time is measured in seconds and is plotted on the log scale. It can be seen that in terms of the training time, the three discriminative parameterizations are many orders of magnitude slower than plain naive Bayes. To show the comparison of the training time of $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$, $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ only, we normalize the results with respect to $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$. Results are shown in Figure 6b. It can be seen that $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ is almost twice as slow as compared to $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$, whereas, $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ is order of magnitude slower that $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$.



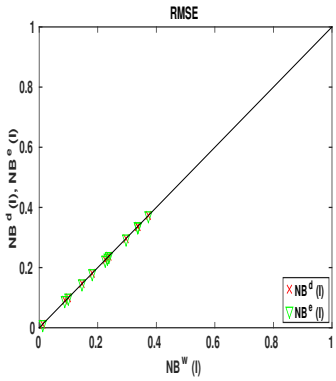
(a) 0-1 Loss on *All* datasets



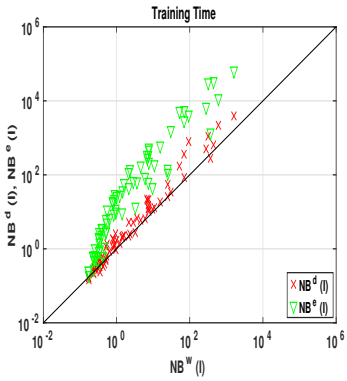
(b) 0-1 Loss on *Big* datasets



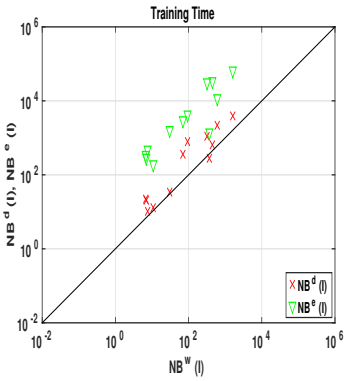
(c) RMSE on *All* datasets



(d) RMSE on *Big* datasets



(e) Training time on *All* datasets



(f) Training time on *Big* datasets

Fig. 4: Comparative scatter of results for NB^w, NB^d and NB^e. NB^w is on the X-axis whereas NB^d (red-cross) and NB^e (green-triangle) are on the Y-axis. For any points above the diagonal line NB^w wins.

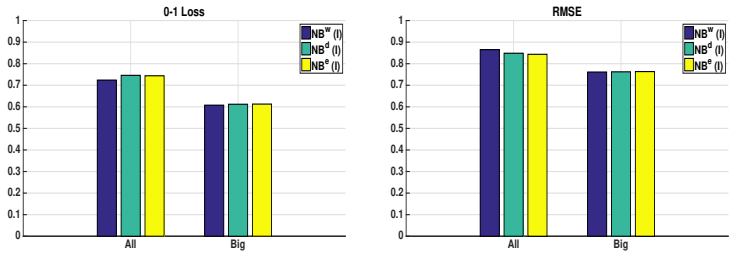
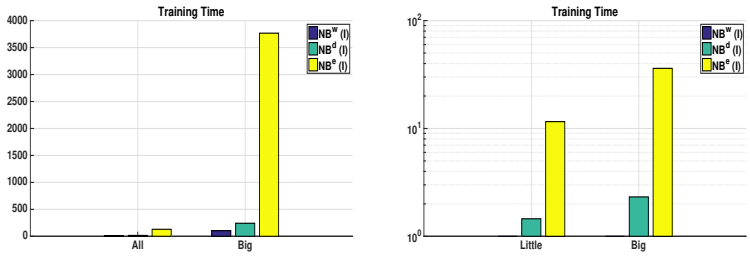


Fig. 5: Geometric mean of 0-1 Loss and RMSE for NB, NB^w, NB^d and NB^e on *All* and *Big* datasets. Results are normalized with respect to NB.



(a) Training Time (b) Training Time

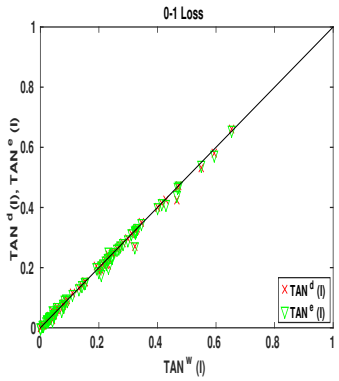
Fig. 6: (Figure: 6a) Geometric mean of training time for NB, NB^w, NB^d and NB^e on *All* and *Big* datasets. Results are normalized with respect to NB. (Figure: 6b) Geometric mean of training time for NB^w, NB^d and NB^e on *All* and *Big* datasets. Results are normalized with respect to NB^w.

7.2 TAN Structure

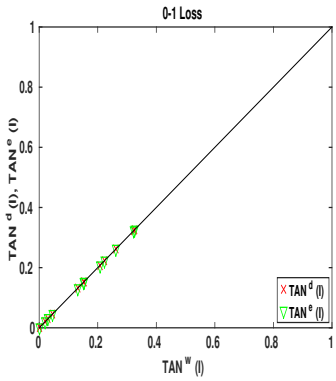
Figure 7 shows the comparative spread of 0-1 Loss, RMSE and training time in seconds of TAN^{w}, TAN^{d} and TAN^{e} on *All* and *Big* datasets. A trend similar to that of NB can be seen. With a similar spread of 0-1 Loss and RMSE among the three parameterizations, training time is greatly improved for TAN^{w} when compared with TAN^{d} and TAN^{e}. Note, as pointed out before, that minor variation in the performance of three discriminative parameterizations is due to the numerical issues with-in the solver on some small datasets. On *big* datasets, one can see a similar spread of 0-1 Loss and RMSE.

The geometric means of the 0-1 Loss and RMSE results are shown in Figure 8. It can be seen that TAN^{d}, TAN^{e} and TAN^{w}, on average results in much better accuracy than generative model (TAN).

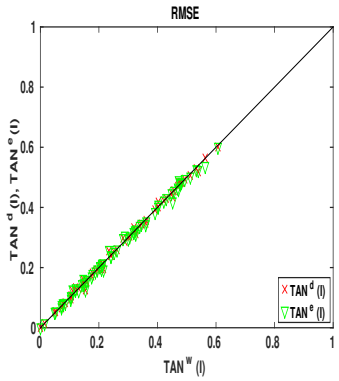
A comparison of the training time is shown in Figure 9a. It can be seen that, like NB, training time of the discriminative methods is orders of magnitude longer than that of generative learning. Note, the training time of discriminative learning also includes the structure learning process. We also show the comparison of the training time of TAN^{d}, TAN^{e} and TAN^{w} in Figure 9b. Like, NB, it can be seen that TAN^{w} is almost twice as fast as TAN^{d}, whereas, TAN^{e} is orders of magnitude slower than TAN^{w}.



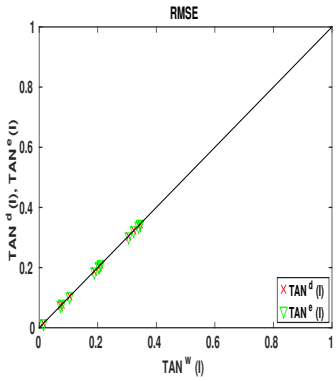
(a) 0-1 Loss on *All* datasets



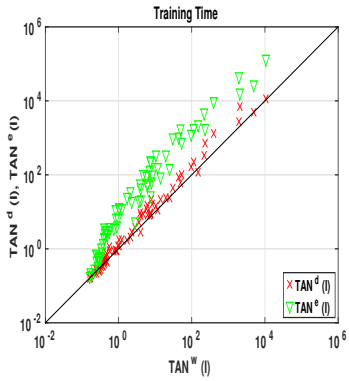
(b) 0-1 Loss on *Big* datasets



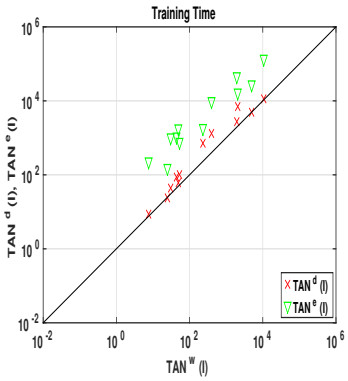
(c) RMSE on *All* datasets



(d) RMSE on *Big* datasets



(e) Training time on *All* datasets



(f) Training on *Big* datasets

Fig. 7: Comparative scatter of results for TAN^w, TAN^d and TAN^e. TAN^w is on the X-axis whereas TAN^d (red-cross) and TAN^e (green-triangle) are on the Y-axis. For any points above the diagonal line TAN^w wins.

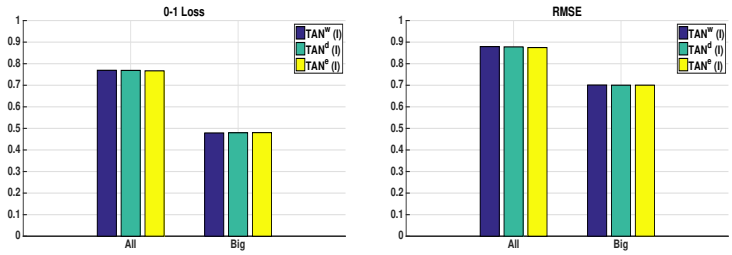
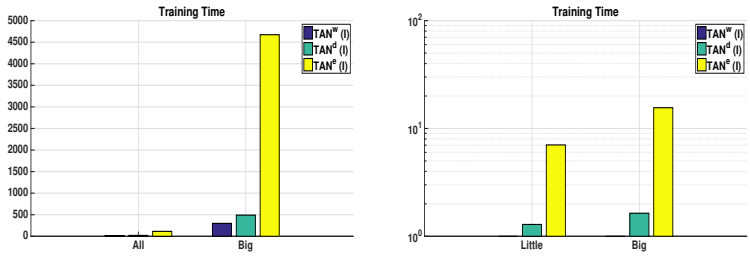


Fig. 8: Geometric mean of 0-1 Loss and RMSE for TAN^{w}, TAN^{d} and TAN^{e} on *All* and *Big* datasets. Results are normalized with respect to TAN.



(a) Training Time (b) Training Time

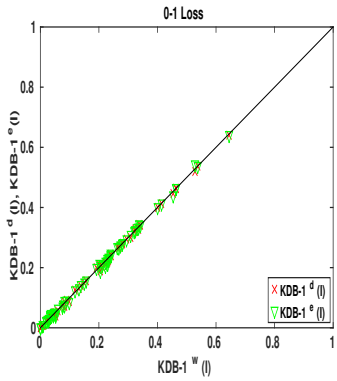
Fig. 9: (Figure: 9a) Geometric mean of training time for TAN, TAN^{w}, TAN^{d} and TAN^{e} on *All* and *Big* datasets. Results are normalized with respect to NB. (Figure: 9b) Geometric mean of training time for TAN^{w}, TAN^{d} and TAN^{e} on *All* and *Big* datasets. Results are normalized with respect to TAN^{w}.

7.3 KDB ($K = 1$) Structure

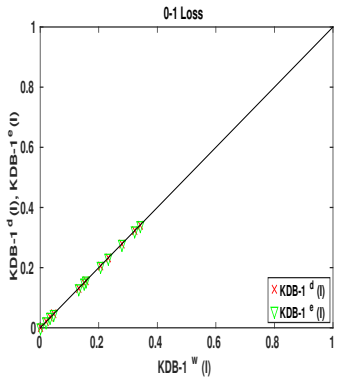
Figure 10 shows the comparative spread of 0-1 Loss, RMSE and training time in seconds of KDB-1^{w}, KDB-1^{d} and KDB-1^{e} on *All* and *Big* datasets. Like NB and TAN, it can be seen that a similar spread of 0-1 Loss and RMSE is present among the three parameterizations of discriminative learning. Similarly, it can be seen that training time is greatly improved for KDB-1^{w} when compared with KDB-1^{d} and KDB-1^{e}.

Geometric average of the 0-1 Loss and RMSE results are shown in Figure 11. It can be seen that the three discriminative parameterizations have better 0-1 Loss and RMSE than generative learning (KDB-1).

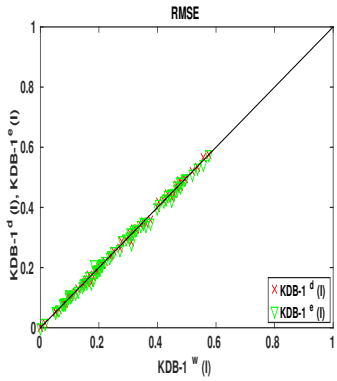
A comparison of the training time is given in Figure 12a. Note, the training time of discriminative methods also includes the time of structure learning. It can be seen that discriminative learning leads to a significantly longer training time than generative learning. We compare the training time of KDB-1^{d}, KDB-1^{e} and KDB-1^{w} in Figure 12b. Like, NB and TAN structure, it can be seen that KDB-1^{w} is faster than both KDB-1^{d} and KDB-1^{e}.



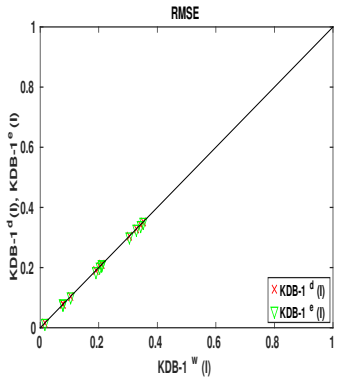
(a) 0-1 Loss on *All* datasets



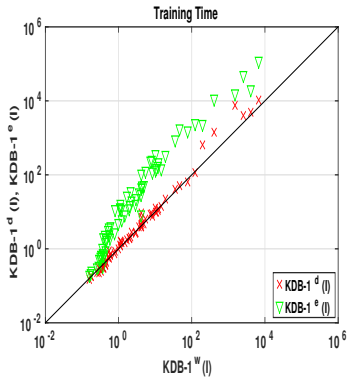
(b) 0-1 Loss on *Big* datasets



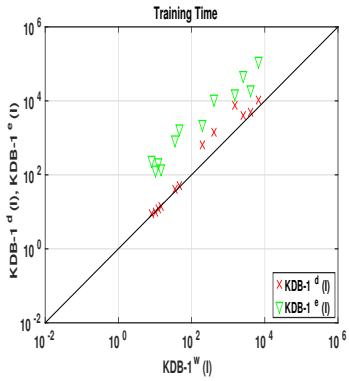
(c) RMSE on *All* datasets



(d) RMSE on *Big* datasets



(e) Training time on *All* datasets



(f) Training on *Big* datasets

Fig. 10: Comparative scatter of results for KDB-1^{w}, KDB-1^{d} and KDB-1^{e}. KDB-1^{w} is on the X-axis whereas KDB-1^{d} (red-cross) and KDB-1^{e} (green-triangle) are on the Y-axis. For any points above the diagonal line KDB-1^{w} wins.

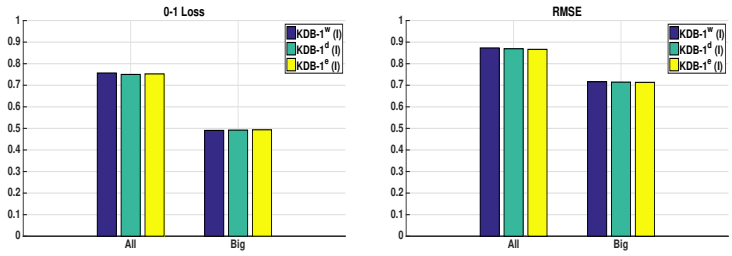
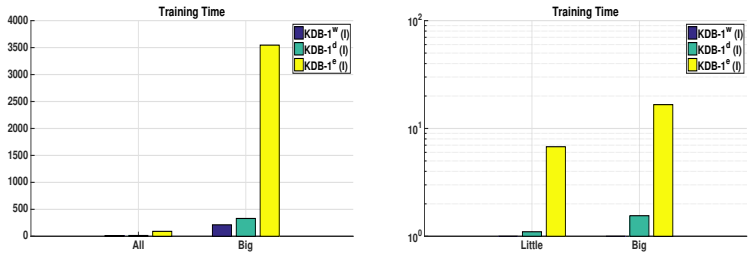


Fig. 11: Geometric mean of training time and RMSE for KDB-1^{w}, KDB-1^{d} and KDB-1^{e} on *All* and *Big* datasets. Results are normalized with respect to KDB-1.



(a) Training Time (b) Training Time

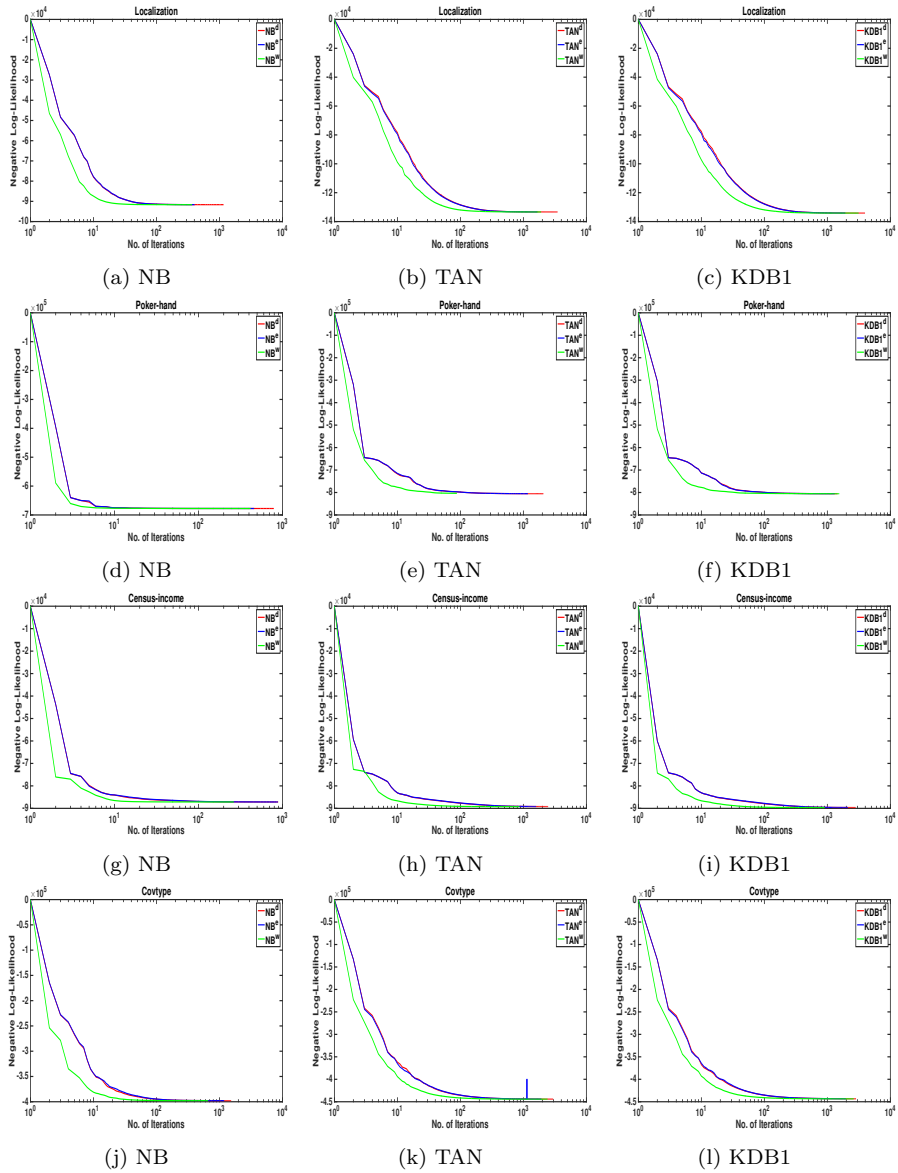
Fig. 12: (Figure: 12a) Geometric mean of training time for KDB1, KDB-1^{w}, KDB-1^{d} and KDB-1^{e} on *All* and *Big* datasets. Results are normalized with respect to NB. (Figure: 12b) Geometric mean of training time for KDB-1^{w}, KDB-1^{d} and KDB-1^{e} on *All* and *Big* datasets. Results are normalized with respect to KDB-1^{w}.

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 13 and 14.

As discussed in Section 5.1, in Figure 13, parameters are initialized to zero, whereas, in Figure 14, 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 72 datasets. A comparison on many more datasets is given in Figure 18 and 19 in Appendix B.

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 in the fifth iteration. If the three parameterizations follow similar convergence, one should expect many zeros in the histogram. Note that if after the 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 the 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 15, 16 and 17 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.



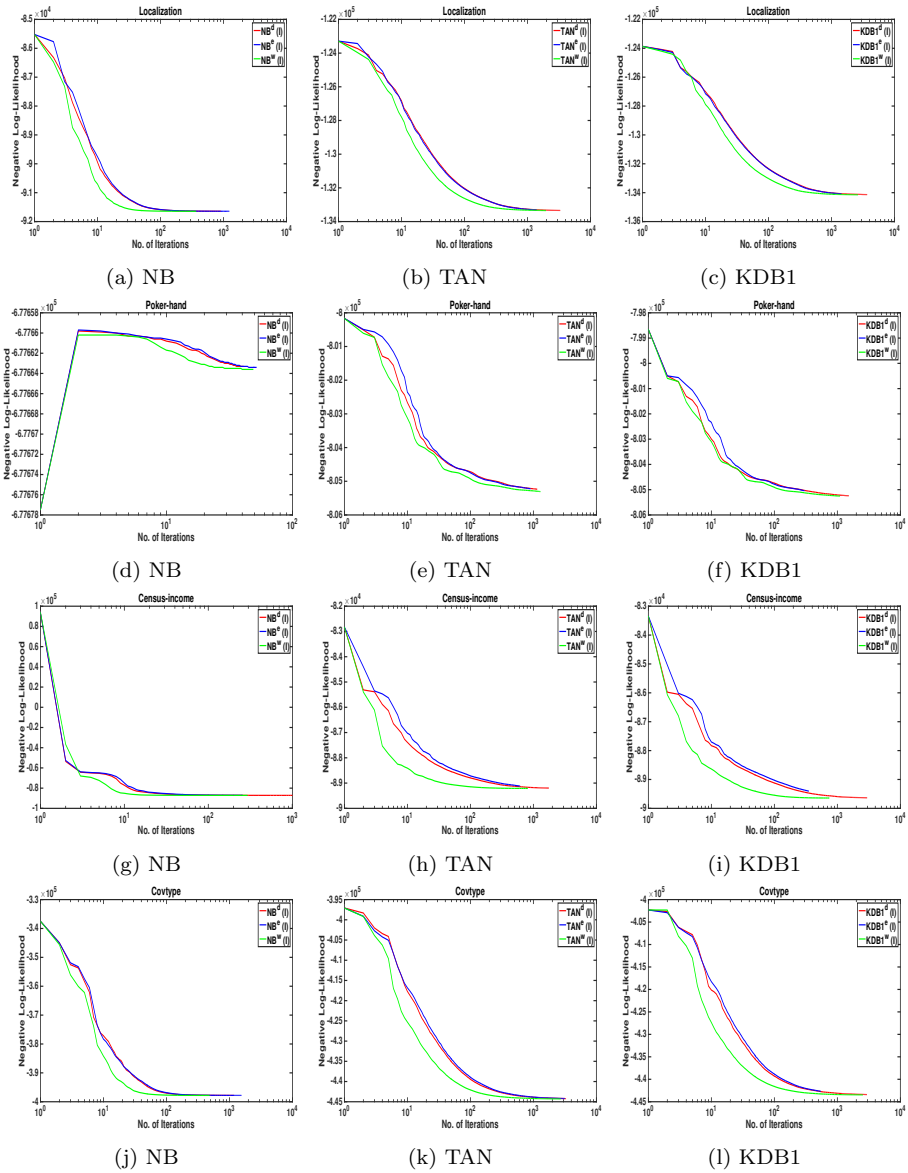
(a) NB (b) TAN (c) KDB1

(d) NB (e) TAN (f) KDB1

(g) NB (h) TAN (i) KDB1

(j) NB (k) TAN (l) KDB1

Fig. 13: Comparison of rate of convergence on the four biggest datasets for NB, TAN and KDB ($K = 1$) (right column) structures. The X-axis is on log scale. Parameters are initialized to zero. Note, the first iteration is actually NLL before the start of optimization. It can be seen that the three parameterizations start from the same point in the space.



(a) NB (b) TAN (c) KDB1

(d) NB (e) TAN (f) KDB1

(g) NB (h) TAN (i) KDB1

(j) NB (k) TAN (l) KDB1

Fig. 14: Comparison of rate of convergence on the four biggest datasets for NB, TAN and KDB ($K = 1$) (right column) structures. The X-axis is on log scale. Parameters are initialized to the log of the MAP estimates. Note, the first iteration is actually NLL before the start of optimization. It can be seen that the three parameterizations start from the same point in the space.

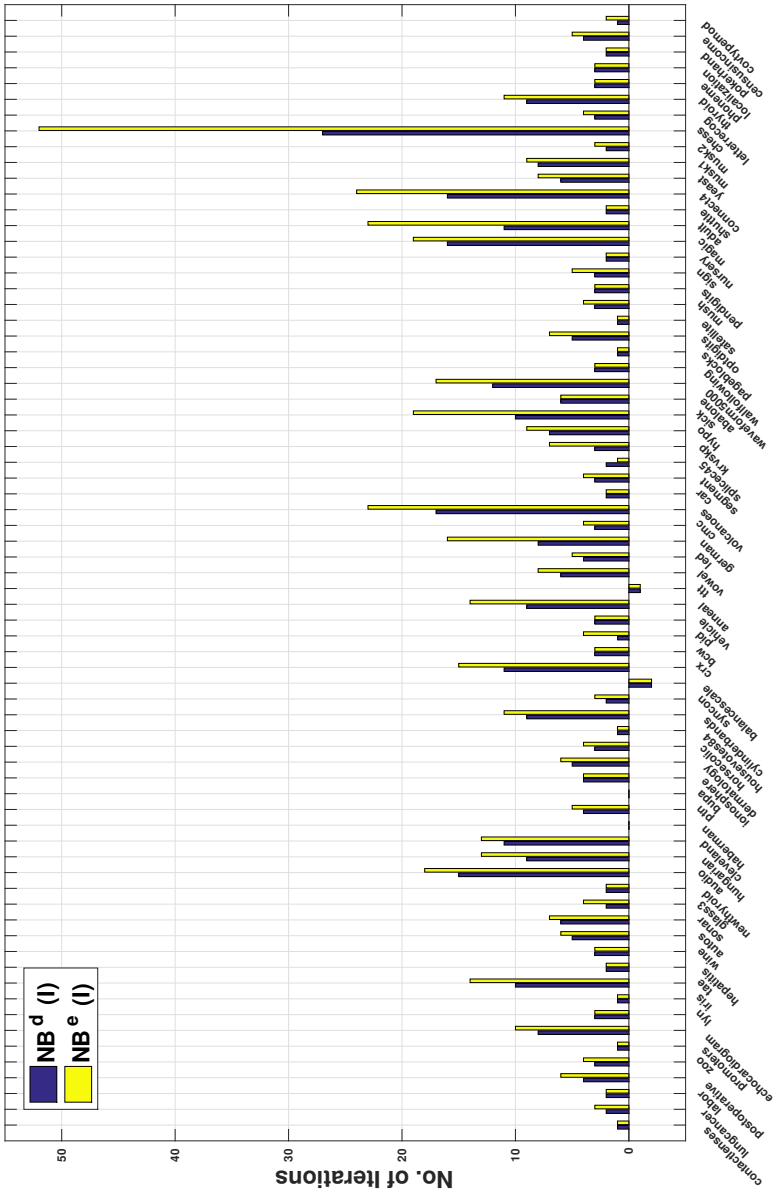


Fig. 15: Number of iterations (after 5 iterations), it takes NB^d and NB^e to reach NLL that NB^w achieved after 5 iterations. If NLL of NB^w is less that that of NB^d or NB^e, (negative of the) number of iterations it takes NB^w to reach that of NB^d and that of NB^e (denoted as NB^{w-e}) are plotted with negative sign.

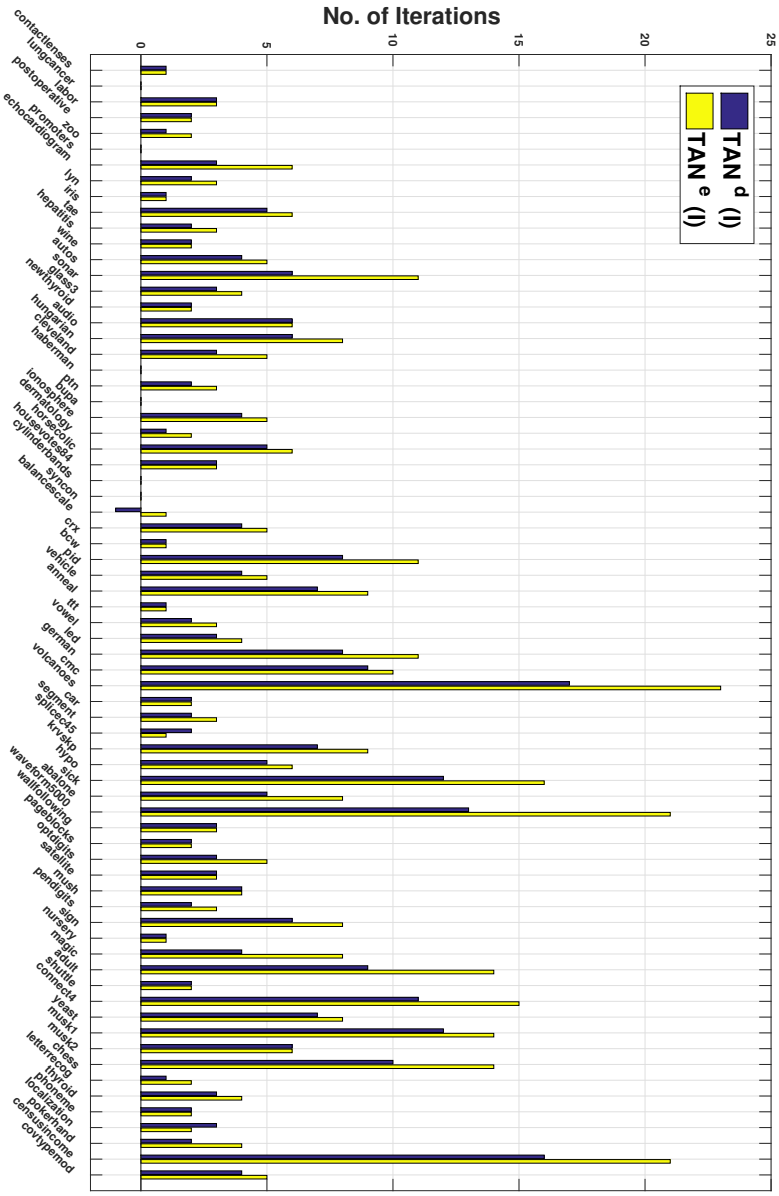


Fig. 16: Number of iterations (after 5 iterations), it takes TAN^d and TAN^e to reach NLL that TAN^w achieved after 5 iterations. If NLL of TAN^w is less that that of TAN^d or TAN^e, (negative of the) number of iterations it takes TAN^w to reach that of TAN^d and that of TAN^e (denoted as $TAN^{w\text{-}e}$) are plotted with negative sign.

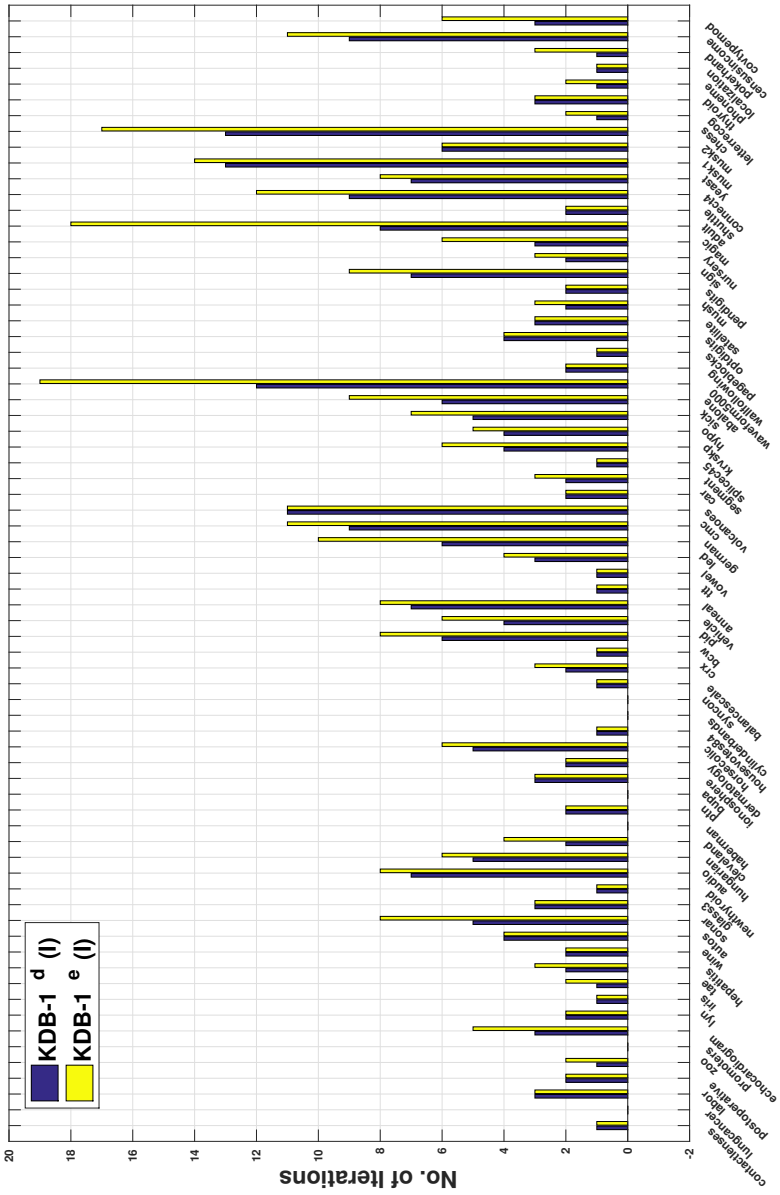


Fig. 17: Number of iterations (after 5 iterations), it takes KDB-1^d and KDB-1^e to reach NLL that KDB-1^w achieved after 5 iterations. If NLL of KDB-1^w is less that that of KDB-1^d or KDB-1^e, (negative of the) number of iterations it takes KDB-1^w to reach that of KDB-1^d and that of KDB-1^e (denoted as KDB$1^{w\text{-}e}$) are plotted with negative sign.

	NB^{w} vs. NB		**TAN^{w} vs. TAN**		**KDB-1^{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 4: Win-Draw-Loss: NB^{w} vs. NB, TAN^{w} vs. TAN and KDB-1^{w} vs. KDB-1. Significant results are shown in bold.

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 4, we compare the 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 $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$, $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ 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, if TAN or KDB classifiers have not encountered a $<$ `feature-value, parent-value, class-value` $>$ combination at training time they can resort back to $<$ `feature-value, class-value` $>$ at testing time. For instance TAN classifier can step back to NB and NB can step back to class prior probabilities. Such elegantly back-tracking is missing from discriminative learning. If a certain combination does not exist in the data, parameters associated to that combination 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 them if they occur in the testing data. How to incorporate such hierarchical 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 structures that the proposed weighted discriminative 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. As discussed in Section 5.4, 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 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.
- Handling 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 [22], 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 [7]. Code with running instructions can be download from `https://github.com/nayyarzaidi/EBNC.git`.

10 Acknowledgments

This research has been supported by the Australian Research Council (ARC) under grant DP140100087. This material is based upon work supported by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-15-1-4007 and FA2386-16-1-4023.

The authors would like to thank Jesús Cerquides, Reza Haffari and Ana Martinez for helpful discussion during the course of this research.

A Proof of Theorem 1

Let us use Lagrange multipliers for constraints in Equation 4 to be placed in Equation 3. Now, we can maximize the resulting objective function:

$$\mathrm{LL}(\mathcal{B}) + \lambda_0(1 - \sum_{y \in \mathcal{Y}} \theta_{y|\Pi_{0(\mathbf{x})}}) + \sum_{i}^{n} \lambda_i (1 - \sum_{x_i \in \mathcal{X}_i} \theta_{x_i|\Pi_i(\mathbf{x})}),$$

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 4, we get:

$$\sum_{x_i \in \mathcal{X}_i} \frac{\sum_{j=1}^{N} N_{x_i,y,\Pi_i(\mathbf{x})}}{\lambda_i} = 1,$$

which implies: $\lambda_i = \sum_{x_i \in \mathcal{X}_i} \sum_{j=1}^{N} N_{x_i,y,\Pi_i(\mathbf{x})}$. 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: $\mathrm{P}_{\mathcal{D}}(x_i|\Pi_i(\mathbf{x}))$.

B Convegence Curves

Continued from Section 7.4, in this section, we present some more results to compare the convergences of three discriminative parameterizations. In Figure 18, we initialize the parameterizations with the generative estimates, whereas, in Figure 19, parameters are initialized to zero.

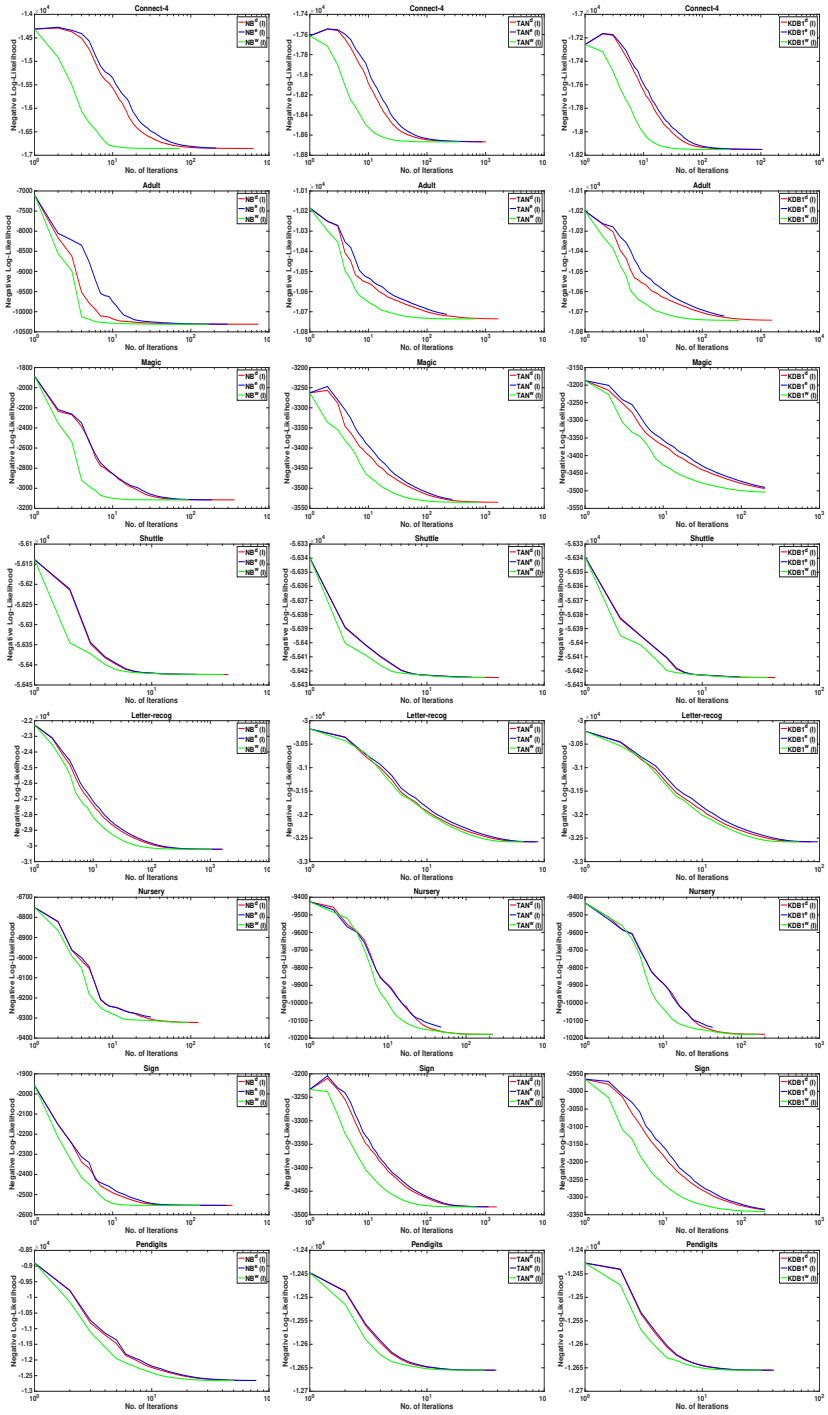


Fig. 18: Comparison of rate of convergence on the six sample datasets for NB (left column), TAN (middle column) and KDB ($K = 1$) (right row) structures. The X-axis is on log scale. Parameters are initialized to the log of the MAP estimates.

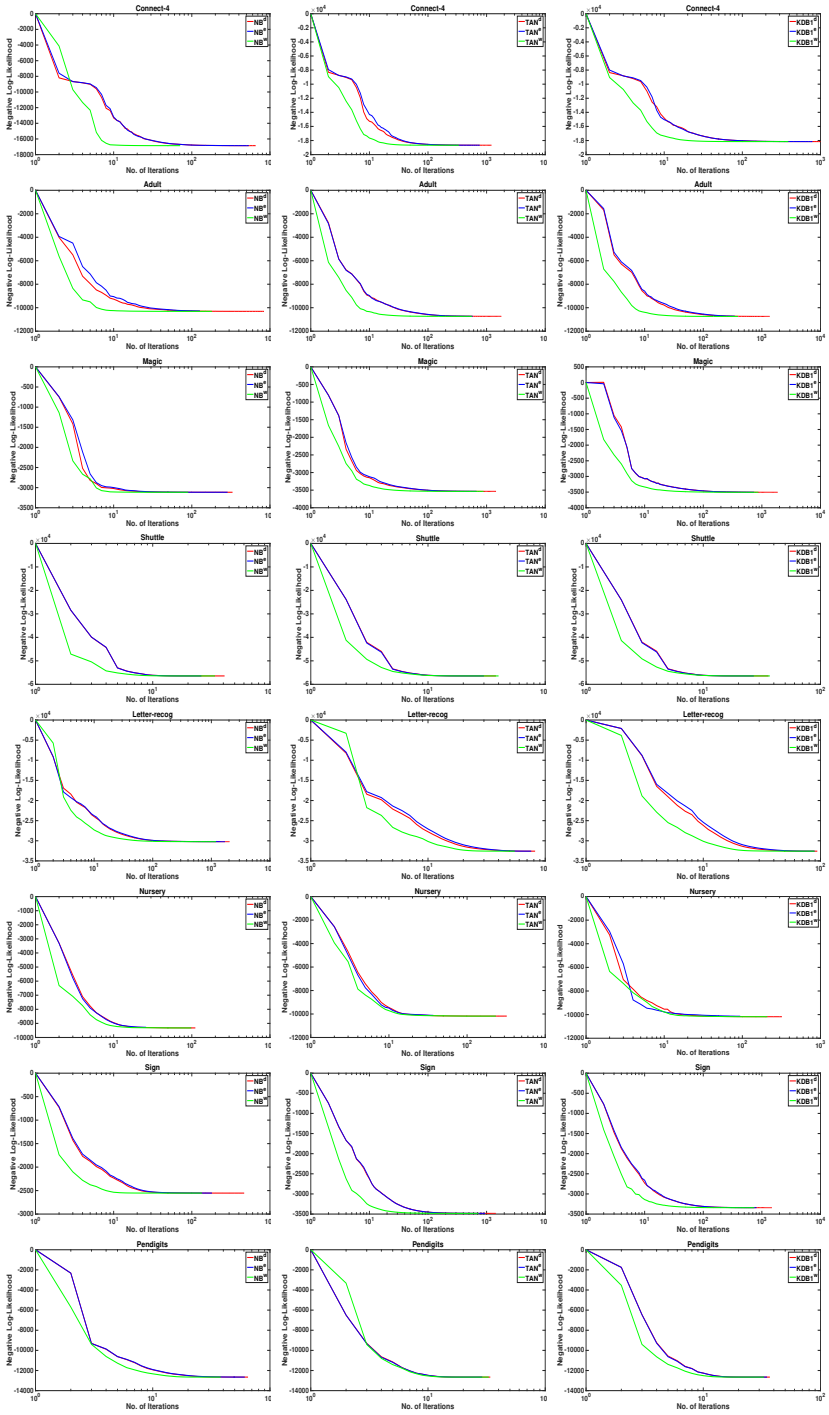


Fig. 19: Comparison of rate of convergence on the four biggest datasets for NB (left column), TAN (middle column) and KDB ($K = 1$) (right row) structures. The X-axis is on log scale. Parameters are initialized to the log of the MAP estimates.

C Training Time Significance Test

Let us discuss the significance of the training time for three discriminative parameterizations using Friedman and Nemenyi tests. Following procedure is taken to generate the results:

- We have three algorithms to compare that is: $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$, $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$, therefore, $k = 3$.
- We compare the results on 72 datasets, therefore, $N = 72$.
- Friedman test rank each algorithm for each dataset separately. In case of ties, it uses average ranks.
- If r_i^j is the rank of algorithm j on i-th dataset, average rank for each algorithm compared are computed as: $R_j = \frac{1}{N}\sum_i r_i^j$.
- We state:
  - Null hypothesis – Algorithms are equivalent and, therefore, ranks should be equal. Mean rank is 2.5.
  - p-value – probability of getting ranks R_j if null-hypothesis as stated in previous point is true.
- Compute the Friedman statistics:

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right],$$

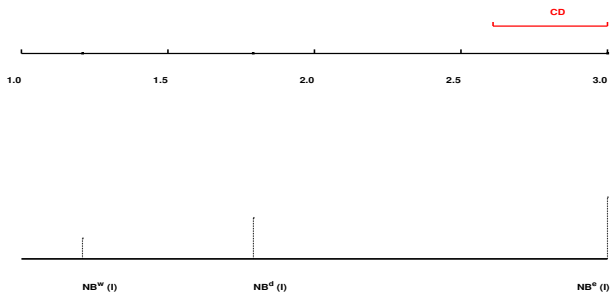
  to determine if the measure ranks are significantly different from the mean rank of 2.5 (under null hypothesis).
- If the p-value is ≤ 0.05, we reject the null hypothesis and proceed with the post-hoc test.
- We use the Nemenyi test which states that the performance of two algorithms is significantly different if the corresponding average ranks differ by at least the critical difference (CD) of:

$$\mathrm{CD} = q_\alpha\sqrt{\frac{k(k+1)}{6N}},$$

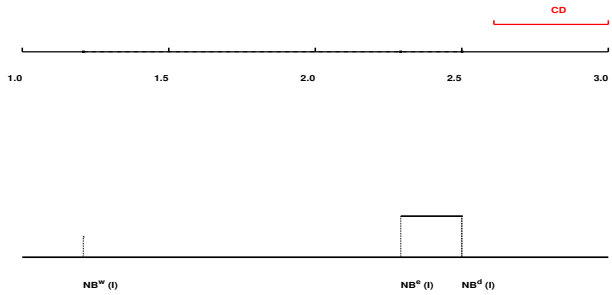
  where q_α in our experiments is 2.3430 as $k = 3$.
- If the difference between top rank and the bottom rank is less than the CD, we conclude pos-hoc test to be not powerful.
- Otherwise (following the graphical representation of [5],), we plot the ranks (along with the name of algorithm) on a horizontal line. Algorithms are connected by a line if their differences are not significant. We also show the CD on the same scale to highlight the significance of the difference of two ranks.

We show the significance test using Friedman and Nemenyi test on *All* datasets in terms of training time and no. of iterations it takes each algorithm to converge in Figures 20, 21 and 22. It can be seen that for all three structures that is NB, TAN and KDB-1, $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ is rank lower than $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ both in terms of training time and no. of iterations.

For NB and TAN, $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ has significantly better training time and converges in far fewer iterations than the other two. However, for KDB-1 structure, the difference is not significant between $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ and $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$.

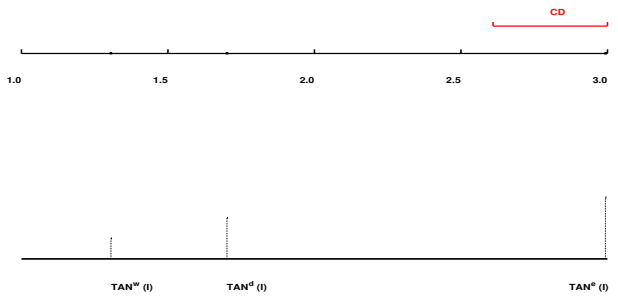


(a) Training Time – $p \leq 0.001$, CD = 0.39

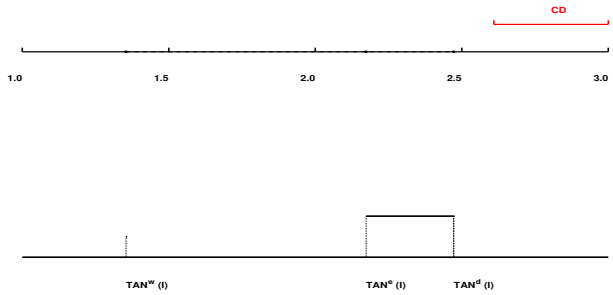


(b) No. of Iterations – $p \leq 0.001$, CD = 0.39

Fig. 20: Significance testing of Training time (Figure: 20a) and No. of Iterations (Figure: 20b) to converge for three parameterizations NB^w, NB^d and NB^e, with Friedman and Nemenyi test on *All* datasets. Ranks are different according to Friedman test and, therefore, null-hypothesis is rejected. (Nemenyi) Post-hoc test is powerful.

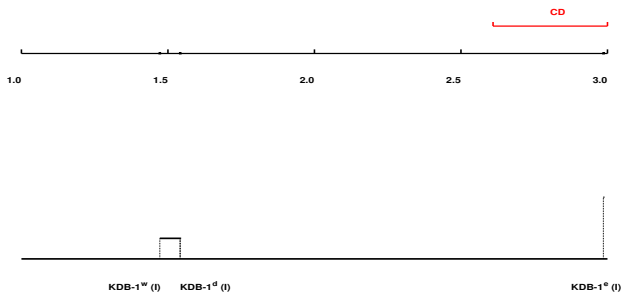


(a) Training Time – $p \leq 0.001$, CD = 0.39

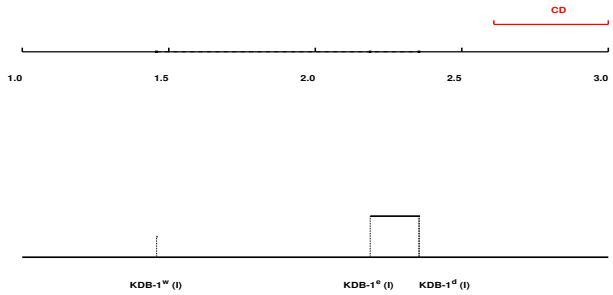


(b) No. of Iterations – – $p \leq 0.001$, CD = 0.39

Fig. 21: Significance testing of Training time (Figure: 21a) and No. of Iterations (Figure: 21b) to converge for three parameterizations TAN^{w}, TAN^{d} and TAN^{e}, with Friedman and Nemenyi test on *All* datasets. Ranks are different according to Friedman test and, therefore, null-hypothesis is rejected. (Nemenyi) Post-hoc test is powerful.



(a) Training Time – $p \leq 0.001$, CD = 0.39



(b) No. of Iterations – $p \leq 0.001$, CD = 0.39

Fig. 22: Significance testing of Training time (Figure: 22a) and No. of Iterations (Figure: 22b) to converge for three parameterizations KDB-1^{w}, KDB-1^{d} and KDB-1^{e}, with Friedman and Nemenyi test on *All* datasets. Ranks are different according to Friedman test and, therefore, null-hypothesis is rejected. (Nemenyi) Post-hoc test is powerful.

D Convergence Significance Test

We compare the NLL obtained by each parameterization at fifth,tenth and fiftieth iteration and presented results in terms of win-draw-loss in Table 5 for $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ versus $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ and for $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ versus $\mathcal{M}_{\mathrm{e}}^{\mathcal{B}^*}$ in Table 6. It can be seen from the two tables, that $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ wins significantly against $\mathcal{M}_{\mathrm{d}}^{\mathcal{B}^*}$ with all three structures. The trend is extremely impressive when comparing against 12 big datasets. Since, each iteration encompasses looping through all the data, these are datasets where each iteration is expensive. It can be seen that $\mathcal{M}_{\mathrm{w}}^{\mathcal{B}^*}$ achieves a better NLL not only after fifth and tenth iteration, but better even after fiftieth iteration.

	NB^{w} vs. NB^{d}		**TAN^{w} vs. TAN^{d}**		**KDB-1^{w} vs. KDB-1^{d}**	
	W-D-L	p	W-D-L	p	W-D-L	p
			All Datasets			
NLL (5)	67/2/2	**<0.001**	64/6/1	**<0.001**	66/5/0	**<0.001**
NLL (10)	67/2/2	**<0.001**	63/7/1	**<0.001**	65/5/1	**<0.001**
NLL (50)	56/14/1	**<0.001**	44/25/2	**<0.001**	46/22/3	**<0.001**
			Big Datasets			
NLL (5)	12/0/0	**<0.001**	12/0/0	**<0.001**	12/0/0	**<0.001**
NLL (10)	12/0/0	**<0.001**	12/0/0	**<0.001**	12/0/0	**<0.001**
NLL (50)	12/0/0	**<0.001**	11/1/0	**<0.001**	11/1/0	**<0.001**

Table 5: Win-Draw-Loss: NB^{w} vs. NB^{d}, TAN^{w} vs. TAN^{d} and KDB-1^{w} vs. KDB-1^{d}. Significant results are shown in bold.

	NB^{w} vs. NB^{e}		**TAN^{w} vs. TAN^{e}**		**KDB-1^{w} vs. KDB-1^{e}**	
	W-D-L	p	W-D-L	p	W-D-L	p
			All Datasets			
NLL (5)	69/0/2	**<0.001**	67/4/0	**<0.001**	66/5/0	**<0.001**
NLL (10)	68/0/3	**<0.001**	65/5/1	**<0.001**	65/5/1	**<0.001**
NLL (50)	61/10/0	**<0.001**	51/19/1	**<0.001**	52/19/0	**<0.001**
			Big Datasets			
NLL (5)	12/0/0	**<0.001**	12/0/0	**<0.001**	12/0/0	**<0.001**
NLL (10)	12/0/0	**<0.001**	12/0/0	**<0.001**	12/0/0	**<0.001**
NLL (50)	12/0/0	**<0.001**	11/1/0	**<0.001**	11/1/0	**<0.001**

Table 6: Win-Draw-Loss: NB^{w} vs. NB^{e}, TAN^{w} vs. TAN^{e} and KDB-1^{w} vs. KDB-1^{e}. Significant results are shown in bold.

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. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. Information Theory, IEEE Transactions on **14**(3), 462–467 (1968)
5. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**, 1–30 (2006)
6. Fayyad, U.M., Irani, K.B.: On the handling of continuous-valued attributes in decision tree generation. Machine Learning **8**(1), 87–102 (1992)
7. Frank, A., Asuncion, A.: UCI machine learning repository (2010). URL `http://archive.ics.uci.edu/ml`
8. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning **29**(2), 131–163 (1997)
9. Greiner, R., Zhou, W.: Structural extension to logistic regression: Discriminative paramter learning of belief net classifiers. In: AAAI (2002)
10. 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)
11. Grossman, D., Domingos, P.: Learning bayesian network classifiers by maximizing conditional likelihood. In: ICML (2004)
12. Heckerman, D., Meek, C.: Models and selection criteria for regression and classification. In: International Conference on Uncertainity in Artificial Intelligence (1997)
13. Jebara, T.: Machine Learning: Discriminative and Generative. Springer International Series (2003)
14. Kohavi, R., Wolpert, D.: Bias plus variance decomposition for zero-one loss functions. In: ICML, pp. 275–283 (1996)
15. Langford, J., Li, L., Strehl, A.: Vowpal wabbit online learning project (2007)
16. Martinez, A., Chen, S., Webb, G.I., Zaidi, N.A.: Scalable learning of bayesian network classifiers. Journal of Machine Learning Research (2015)
17. 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)
18. Pernkopf, F., Bilmes, J.: Discriminative versus generative parameter and structure learning of bayesian network classifiers. In: ICML (2005)
19. Pernkopf, F., Bilms, J.A.: Efficient heuristics for discriminative structure learning of bayesian network classifiers. Journal of Machine Learning Research (2010)
20. Pernkopf, F., Wohlmayr, M.: On discriminative parameter learning of bayesian network classifiers. In: ECML PKDD (2009)
21. Ripley, B.D.: Pattern Recognition and Neural Networks. Cambridge University Press (1996)
22. 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)
23. Rubinstein, Y.D., Hastie, T.: Discriminative vs informative learning. In: AAAI (1997)
24. Sahami, M.: Learning limited dependence bayesian classifiers. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp. 335–338 (1996)
25. Su, J., Zhang, H., Ling, C., Matwin, S.: Discriminative parameter learning for bayesian networks. In: ICML (2008)
26. Webb, G.I.: Multiboosting: A technique for combining boosting and wagging. Machine Learning **40**(2), 159–196 (2000)
27. 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)
28. 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)

29. 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)
30. Zaidi, N.A., Petitjean, F., Webb, G.I.: Preconditioning an artificial neural network using naive bayes. In: Proceedings of the 20th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD) (2016)
31. Zaidi, N.A., Webb, G.I., Carman, M.J., Petitjean, F.: Deep broad learning - Big models for Big data. arXiv:1509.01346 (2015)
32. 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)