Discriminative Scoring of Bayesian Network Classifiers: a Comparative Study

Ad Feelders and Jevgenijs Ivanovs
Department of Information and Computing Science
Universiteit Utrecht, The Netherlands

Abstract

We consider the problem of scoring Bayesian Network Classifiers (BNCs) on the basis of the conditional loglikelihood (CLL). Currently, optimization is usually performed in BN parameter space, but for perfect graphs (such as Naive Bayes, TANs and FANs) a mapping to an equivalent Logistic Regression (LR) model is possible, and optimization can be performed in LR parameter space. We perform an empirical comparison of the efficiency of scoring in BN parameter space, and in LR parameter space using two different mappings. For each parameterization, we study two popular optimization methods: conjugate gradient, and BFGS. Efficiency of scoring is compared on simulated data and data sets from the UCI Machine Learning repository.

1 Introduction

Discriminative learning of Bayesian Network Classifiers (BNCs) has received considerable attention recently (Greiner et al., 2005; Pernkopf and Bilmes, 2005; Roos et al., 2005; Santafé et al., 2005). In discriminative learning, one chooses the parameter values that maximize the *conditional* likelihood of the class label given the attributes, rather then the *joint* likelihood of the class label and the attributes. It is well known that conditional loglikelihood (CLL) optimization, although arguably more appropriate in a classification setting, is computationally more expensive because there is no closed-form solution for the ML estimates and therefore numerical optimization techniques have to be applied. Since in structure learning of BNCs many models have to be scored, the efficiency of scoring a single model is of considerable interest.

For BNCs with perfect independence graphs (such as Naive Bayes, TANs, FANs) a mapping to an equivalent Logistic Regression (LR) model is possible, and optimization can be performed in LR parameter space. We consider two such mappings: one proposed in (Roos et al., 2005), and a different mapping, that, although relatively straightforward, has to our knowledge not been proposed before in discriminative learning of BNCs. We conjecture that scoring models in LR space using our proposed mapping is more efficient than scoring in BN space, because the logistic regression model has fewer parameters than its BNC counterpart and because the LR model is known to have a strictly concave loglikelihood function. To test this hypothesis we perform experiments to compare the efficiency of model fitting with both LR parameterizations, and the more commonly used BN parameterization.

This paper is structured as follows. In section 2 we introduce the required notation and basic concepts. Next, in section 3 we describe two mappings from BNCs with perfect graphs to equivalent LR models. In section 4 we give a short description of the optimization methods used in the experiments, and motivate their choice. Subsequently, we compare the efficiency of discriminative learning in LR parameter space and BN parameter space, using the optimization methods discussed. Finally, we present the conclusions in section 7.

2 Preliminaries

2.1 Bayesian Networks

We use uppercase letters for random variables and lowercase for their values. Vectors are written in boldface. A Bayesian network (BN) $(\mathbf{X}, G = (V, E), \boldsymbol{\theta})$ consists of a discrete random vector $\mathbf{X} = (X_0, \ldots, X_n)$, a directed acyclic graph (DAG) G representing the directed independence graph of $\mathbf{X}$,

and a set of conditional probabilities (parameters) $\boldsymbol{\theta}$. $V = \{0, 1, \ldots, n\}$ is the set of nodes of G, and E the set of directed edges. Node i in G corresponds to random variable X_i. With $pa(i)$ ($ch(i)$) we denote the set of parents (children) of node i in G. We write $\mathbf{X}_S, S \subseteq \{0, \ldots, n\}$ to denote the projection of random vector $\mathbf{X}$ on components with index in S. The parameter set $\boldsymbol{\theta}$ consists of the conditional probabilities

$$\theta_{x_i|\mathbf{x}_{pa(i)}} = P(X_i = x_i|\mathbf{X}_{pa(i)} = \mathbf{x}_{pa(i)}), 0 \leq i \leq n$$

We use $\mathcal{X}_i = \{0, \ldots, d_i - 1\}$ to denote the set of possible values of X_i, $0 \leq i \leq n$. The set of possible values of random vector $\mathbf{X}_S$ is denoted $\mathcal{X}_S = \times_{i \in S} \mathcal{X}_i$. We also use $\mathcal{X}_i^- = \mathcal{X}_i \setminus \{0\}$, and likewise $\mathcal{X}_S^- = \times_{i \in S} \mathcal{X}_i^-$

In a BN classifier there is one distinguished variable called the class variable; the remaining variables are called attributes. We use X_0 to denote the class variable; $X_1, \ldots, X_n$ are the attributes. To denote the attributes, we also write $\mathbf{X}_A$, where $A = \{1, \ldots, n\}$. We define $\pi(i) = pa(i) \backslash \{0\}$, the non-class parents of node i, and $\phi(i) = \{i\} \cup \pi(i)$.

Finally, we recall the definition of a *perfect* graph: a directed graph in which all nodes that have a common child are connected is called *perfect*.

2.2 Logistic Regression

The basic assumption of logistic regression (Anderson, 1982) for binary class variable $X_0 \in \{0, 1\}$ is

$$\ln \frac{P(X_0 = 1|\mathbf{Z})}{P(X_0 = 0|\mathbf{Z})} = w_0 + \sum_{i=1}^{k} w_i Z_i, \qquad (1)$$

where the predictors Z_i ($i = 1, \ldots, k$) can be single attributes from $\mathbf{X}_A$, but also functions of one or more attributes from $\mathbf{X}_A$. In words: the log posterior odds are linear in the parameters, not necessarily in the basic attributes.

Generalization to a non-binary class variable $X_0 \in \mathcal{X}_0$ gives

$$\ln \frac{P(X_0 = x_0|\mathbf{Z})}{P(X_0 = 0|\mathbf{Z})} = w_0^{(x_0)} + \sum_{i=1}^{k} w_i^{(x_0)} Z_i, \quad (2)$$

for all $x_0 \in \mathcal{X}_0^-$. This model is often referred to as the multinomial logit model or polychotomous logistic regression model.

It is well known that the loglikelihood function of the logistic regression model is concave and has unique maximum (provided the data matrix Z is of full column rank) attained for finite $\mathbf{w}$ except in two special circumstances described in (Anderson, 1982).

2.3 Log-linear models

Let $G = (V, E)$ be the (undirected) independence graph of random vector $\mathbf{X}$, that is E is the set of edges (i, j) such that whenever (i, j) is not in E, the variables X_i and X_j are independent conditionally on the rest. The log-linear expansion of a graphical log-linear model is

$$\ln P(\mathbf{x}) = \sum_{C \subseteq V} u_C(\mathbf{x}_C)$$

where the sum is taken over all complete subgraphs C of G, and all $\mathbf{x}_C \in \mathcal{X}_C^-$, that is, $u_C(\mathbf{x}_C) = 0$ for $i \in C$ and $x_i = 0$ (to avoid overparameterization). The u-term $u_\varnothing(\mathbf{x})$ is just a constant. It is well known that for BN's with perfect directed independence graph, an equivalent graphical log-linear model is obtained by simply dropping the direction of the edges.

3 Mapping to Logistic Regression

In this section we discuss two different mappings from BNCs with a perfect independence graph to equivalent logistic regression models. Equivalent here means that, assuming $P(\mathbf{X}) > 0$, the BNC and corresponding LR model, represent the same set of conditional distributions of the class variable.

3.1 Mapping of Roos et al.

Roos et al. (Roos et al., 2005) define a mapping from BNCs whose canonical form is a perfect graph, to equivalent LR models. The canonical form is obtained by (1) taking the Markov blanket of X_0 (2) marrying any unmarried parents of X_0. This operation does clearly not change the conditional distribution of X_0. They show that if this canonical form is a perfect graph, then the BNC can be mapped to an equivalent LR model. Their mapping creates an LR model with predictors (and corresponding parameters) as follows

1. $Z_{\mathbf{x}_{pa(0)}} = I(\mathbf{X}_{pa(0)} = \mathbf{x}_{pa(0)})$ with parameter $w^{(x_0)}_{\mathbf{x}_{pa(0)}}$, for $\mathbf{x}_{pa(0)} \in \mathcal{X}_{pa(0)}$.
2. $Z_{\mathbf{x}_{\phi(i)}} = I(\mathbf{X}_{\phi(i)} = \mathbf{x}_{\phi(i)})$ with parameter $w^{(x_0)}_{\mathbf{x}_{\phi(i)}}$, for $i \in ch(0)$ and $\mathbf{x}_{\phi(i)} \in \mathcal{X}_{\phi(i)}$.

For a given BNC with parameter value $\boldsymbol{\theta}$ an equivalent LR model is obtained by putting

$$w^{(x_0)}_{\mathbf{x}_{pa(0)}} = \ln \theta_{x_0|\mathbf{x}_{pa(0)}}, \quad w^{(x_0)}_{\mathbf{x}_{\phi(i)}} = \ln \theta_{x_i|\mathbf{x}_{pa(i)}}$$

3.2 Proposed mapping

Like in the previous section, we start from the canonical graph which is assumed to be perfect. Hence, we obtain an equivalent graphical log-linear model by simply dropping the direction of the edges. We then have

$$\begin{aligned}
&\ln \frac{P(X_0 = x_0|\mathbf{X}_A)}{P(X_0 = 0|\mathbf{X}_A)} \\
&= \ln \frac{P(X_0 = x_0, \mathbf{X}_A)/P(\mathbf{X}_A)}{P(X_0 = 0, \mathbf{X}_A)/P(\mathbf{X}_A)} \\
&= \ln P(X_0 = x_0, \mathbf{X}_A) - \ln P(X_0 = 0, \mathbf{X}_A),
\end{aligned}$$

for $x_0 \in \mathcal{X}_0^-$. Filling in the log-linear expansion for $\ln P(X_0 = x_0, \mathbf{X}_A)$ and $\ln P(X_0 = 0, \mathbf{X}_A)$, we see immediately that u-terms that do not contain X_0 cancel, and furthermore that u-terms with $X_0 = 0$ are constrained to be zero by our identification restrictions. Hence we get

$$\begin{aligned}
&\ln P(X_0 = x_0, \mathbf{x}_A) - \ln P(X_0 = 0, \mathbf{x}_A) = \\
&\quad u_{\{0\}}(X_0 = x_0) + \sum_C u_C(X_0 = x_0, \mathbf{x}_C) = \\
&\qquad\qquad w^{(x_0)}_{\varnothing} + \sum_C w^{(x_0)}_{\mathbf{x}_C}
\end{aligned}$$

where C is any complete subgraph of G not containing X_0. Hence, to map to a LR model, we create variables

$$I(\mathbf{X}_C = \mathbf{x}_C), \qquad \mathbf{x}_C \in \mathcal{X}_C^-$$

to obtain the LR specification

$$\ln \frac{P(X_0 = x_0|\mathbf{Z})}{P(X_0 = 0|\mathbf{Z})} = w^{(x_0)}_{\varnothing} + \sum_C w^{(x_0)}_{\mathbf{x}_C} I(\mathbf{X}_C = \mathbf{x}_C)$$

This LR specification models the same set of conditional distributions of X_0 as the corresponding undirected graphical model, see for example (Sutton and McCallum, 2006).

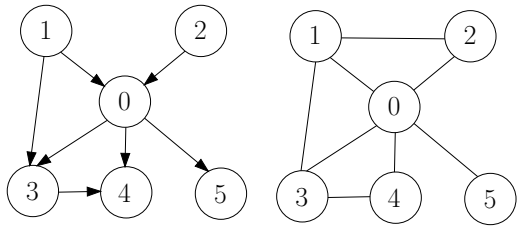

Figure 1: Example BNC (left); undirected graph with same conditional distribution of class (right).

3.3 Example

Consider the BNC depicted in figure 1. Assuming all variables are binary, and using this fact to simplify notation, this maps to the equivalent LR model (with 9 parameters)

$$\begin{aligned}
\ln \frac{P(X_0 = 1|\mathbf{Z})}{P(X_0 = 0|\mathbf{Z})} &= w_{\varnothing} + w_{\{1\}}X_1 + w_{\{2\}}X_2 + \\
&\quad w_{\{3\}}X_3 + w_{\{4\}}X_4 + w_{\{5\}}X_5 + \\
&\quad w_{\{1,2\}}X_1X_2 + w_{\{1,3\}}X_1X_3 + w_{\{3,4\}}X_3X_4
\end{aligned}$$

In the parameterization of (Roos et al., 2005), we map to the equivalent LR_Roos model (with 14 parameters)

$$\begin{aligned}
\ln \frac{P(X_0 = 1|\mathbf{Z})}{P(X_0 = 0|\mathbf{Z})} &= w_{\{1,2\}=(0,0)}Z_{\{1,2\}=(0,0)} + \\
&w_{\{1,2\}=(0,1)}Z_{\{1,2\}=(0,1)} + w_{\{1,2\}=(1,0)}Z_{\{1,2\}=(1,0)} + \\
&w_{\{1,2\}=(1,1)}Z_{\{1,2\}=(1,1)} + w_{\{1,3\}=(0,0)}Z_{\{1,3\}=(0,0)} + \\
&\quad \ldots + w_{\{3,4\}=(1,1)}Z_{\{3,4\}=(1,1)} + \\
&\quad w_{\{5\}=(0)}Z_{\{5\}=(0)} + w_{\{5\}=(1)}Z_{\{5\}=(1)}
\end{aligned}$$

In both cases we set $\mathbf{w}^{(0)} = \mathbf{0}$.

4 Optimization methods

In the experiments, we use two optimization methods: conjugate gradient(CG) and variable metric (BFGS) algorithms (Nash, 1990). Conjugate Gradient is commonly used for discriminative learning of BN parameters, see for example (Greiner et al., 2005; Pernkopf and Bilmes, 2005). In a study of Minka (Minka, 2001), it was shown that CG and BFGS are efficient optimization methods for the logistic regression task.

BFGS is a Hessian-based algorithm, which updates an approximate inverse Hessian matrix of size

r^2 at each step, where r is the number of parameters. CG on the other hand works with a vector of size r. This obviously makes each iteration less costly, however, in general CG exhibits slower convergence in terms of iterations.

Both algorithms at step k compute an update direction $\mathbf{u}_{(k)}$, followed by a line search. This is a one-dimensional search, which looks for a step size α maximizing $f(\alpha) = CLL(\mathbf{w}_{(k)} + \alpha\mathbf{u}_{(k)})$, where $\mathbf{w}_{(k)}$ is a vector of parameter values at step k. It is argued in (Nash, 1990) that neither algorithm benefits from too large a step being taken. Even more, for BFGS it is not desirable that the increase in the function value is different in magnitude from the one determined by the gradient value, $(\mathbf{w}_{(k+1)} - \mathbf{w}_{(k)})^T\mathbf{g}_{(k)}$. Therefore, simple *acceptable point* search is suggested for both methods. In case of CG an additional step is made. Once an acceptable point has been found, we have sufficient information to fit a parabola to the projection of the function on the search direction. The parabola requires three pieces of information: the function value at the end of the last iteration (or the initial point), the projection of the gradient at this point onto the search direction, and the new function value at the acceptable point. If the CLL value at the maximum of the parabola is larger than the CLL value at the acceptable point, then the former becomes the starting point for the next iteration. Another approach to CG line search is Brent's line search method (Press et al., 1992). It iteratively fits a parabola to 3 points. The main difference with the previous approach is that we do find an optimum in the given update direction. This, however, requires more function evaluations at each iteration.

In our experiments we use the implementation of the CG and BFGS methods of the `optim` function of `R` (Venables and Ripley, 2002) which is based on the source code from (Nash, 1990). In addition we implemented CG with Brent's line search (with relative precision for Brent's method set to 0.0002). We refer to this algorithm as CGB. The conjugate gradient method may use different heuristic formulas for computing the update direction. Our preliminary study showed that the difference in performance between these heuristics is small. We use the Polak-Ribiere formula, suggested in (Greiner et al., 2005) for optimization in the BN parameter space.

In our study we compare the rates of convergence for 9 optimization techniques: 3 optimization algorithms (CG, CGB, BFGS) used in 3 parameter spaces (LR, LR_Roos, BN). We compare convergence in terms of (1) iterations of the update direction calculation, and (2) floating point operations (flops). The number of flops provides a fair comparison of the performance of the different methods, but the number of iterations gives us additional insight into the behavior of the methods.

Let $cost_{CLL}$ and $cost_{grad}$ denote the costs in flops of CLL and gradient evaluations respectively, and let $count_{CLL}$ be the number of times CLL is evaluated in a particular iteration. We estimate the cost of one particular iteration of BFGS as $12r^2 + 8r + (4r + cost_{CLL})count_{CLL} + cost_{grad}$; the cost of one CG and CGB iteration is $10r + (4r + cost_{CLL})count_{CLL} + cost_{grad}$. These estimates are obtained by inspecting the source code in (Nash, 1990).

5 Parameter learning

5.1 Logistic regression

Equation 2 can be rewritten as follows

$$P(X_0 = x_0|\mathbf{Z}) = \frac{e^{\mathbf{w}^{(x_0)^T}\mathbf{Z}}}{\sum_{x'_0=0}^{d_0-1} e^{\mathbf{w}^{(x'_0)^T}\mathbf{Z}}},$$

where we put $Z_0 = 1$, which corresponds to the intercept, and fix $\mathbf{w}^{(0)} = \mathbf{0}$. $\mathbf{w}^T$ denotes the transpose of $\mathbf{w}$. The conditional loglikelihood of the parameters given data is

$$CLL(\mathbf{w}) = \log \prod_{i=1}^{N} P(x_0^{(i)}|\mathbf{z}^{(i)})$$
$$= \sum_{i=1}^{N}(\mathbf{w}^{(x_0^{(i)})^T}\mathbf{z}^{(i)} - \log(\sum_{x'_0=0}^{d_0-1} e^{\mathbf{w}^{(x'_0)^T}\mathbf{z}^{(i)}})),$$

where N is the number of observations in the data set. The gradient of the CLL is given by

$$\frac{\partial CLL(\mathbf{w})}{\partial w_k^{(x_0)}} =$$
$$\sum_{i=1}^{N}\left(1_{\{x_0^{(i)}=x_0\}} z_k^{(i)} - \frac{e^{\mathbf{w}^{(x_0)^T}\mathbf{z}^{(i)}} z_k^{(i)}}{\sum_{x'_0=0}^{d_0-1} e^{\mathbf{w}^{(x'_0)^T}\mathbf{z}^{(i)}}}\right)$$

where $x_0 \in \mathcal{X}_0^-$. We note here that the data matrix Z is a sparse matrix of indicators with 1s in

non-zero positions, thus the CLL and gradient functions can be implemented very efficiently. We estimate costs (in flops) according to above formulas: $cost_{CLL} = |Z|(d_0 - 1) + N(d_0 + 2), cost_{grad} = |Z|d_0 + 2Nd_0$, where $|Z|$ denotes number of 1s in the data matrix. Here we used the fact that the multiplication of two vectors $\mathbf{w}^{(x_0)^T}\mathbf{z}^{(i)}$ requires $|\mathbf{z}^{(i)}| - 1$ flops. In case of LR_Roos matrix Z always contains exactly $1 + n - |pa(0)|$ 1s irrespective of graph complexity and dimension of attributes. For our mapping $|Z|$ depends on the sample. In the experiments we used the following heuristic to reduce $|Z|$: for each attribute X_i we code the most frequent value as 0. We note that $|Z|$ is smaller in case of our mapping comparing to LR_Roos mapping, when the structure is not very complex (with regard to the number of parents and the domain size of the attributes) and becomes bigger for more complex structures.

5.2 Bayesian Network Classifiers

Here we follow the approach taken in (Greiner et al., 2005; Pernkopf and Bilmes, 2005). We write

$$\theta^j_{i|k} = P(x_j = i|\mathbf{x}_{pa(j)} = k).$$

We have constraints $\theta^j_{i|k} \geq 0$ and $\sum_{i=0}^{d_j-1} \theta^j_{i|k} = 1$. We reparameterize to incorporate the constraints on $\theta^j_{i|k}$ and use different parameters $\beta^j_{i|k}$ as follows

$$\theta^j_{i|k} = \frac{\exp \beta^j_{i|k}}{\sum_{l=0}^{d_j-1} \exp \beta^j_{l|k}}$$

The CLL is given by

$$CLL(\boldsymbol{\beta}) = \sum_{t=1}^{N} (\log P(\mathbf{x}^{(t)}) - \log \sum_{x_0^{(t)}=0}^{d_0-1} P(\mathbf{x}^{(t)}))$$

Further expansion may be obtained using the factorization of $P(\mathbf{x})$ and plugging in expressions for $\theta^j_{i|k}$. It is easy to see that

$$\frac{\partial \theta^j_{i'|k}}{\partial \beta^j_{i|k}} = \theta^j_{i'|k}(1_{\{i=i'\}} - \theta^j_{i|k}),$$

thus

$$\frac{\partial P(\mathbf{x})}{\partial \beta^j_{i|k}} = 1_{\{\mathbf{x}_{pa(j)}=k\}} P(\mathbf{x})(1_{\{x_j=i\}} - \theta^j_{i|k})$$

Simple calculations result in

$$\frac{\partial CLL(\beta)}{\partial \beta^j_{i|k}} = \sum_{t=1}^{N} \left(1_{\{x_j^{(t)}=i, \mathbf{x}^{(t)}_{pa(j)}=k\}} - 1_{\{\mathbf{x}^{(t)}_{pa(j)}=k\}} \theta^j_{i|k} - \frac{\sum_{x_0^{(t)}=0}^{d_0-1} [P(\mathbf{x}^{(t)})(1_{\{x_j^{(t)}=i, \mathbf{x}^{(t)}_{pa(j)}=k\}} - 1_{\{\mathbf{x}^{(t)}_{pa(j)}=k\}} \theta^j_{i|k})]}{\sum_{x_0^{(t)}=0}^{d_0-1} P(\mathbf{x}^{(t)})} \right)$$

Note that for each t and $j > 1$ only $d_0 d_j$ gradient values are to be considered. In order to obtain $\boldsymbol{\theta}$ from $\boldsymbol{\beta}$ we need $3|\boldsymbol{\beta}|$ flops, where $|\boldsymbol{\beta}|$ denotes the number of components of $\boldsymbol{\beta}$. From the formulas above we estimate $cost_{CLL} = 3|\boldsymbol{\beta}| + N(nd_0 + 2d_0 + 2)$ and $cost_{grad} = 3|\boldsymbol{\beta}| + N(2nd_0 + n + (2d_0 + 1)(3 + d_1 + \ldots + d_n))$.

Finally, we point out that the cost of the gradient is very close to the cost of the CLL in case of LR and is by a factor $2 + 2\overline{d}$ larger in case of BN. This suggests that CGB might be better than CG for BN, but it is very unlikely that it will be better also for LR and LR_Roos parameter spaces. Note that $cost_{CLL}$(BN) is very close to $cost_{CLL}$(LR_Roos), which strongly supports the fairness of our cost estimates.

5.3 Convergence issues

It is well known that LR has a concave loglikelihood function. Since BNCs whose canonical form is a perfect graph are equivalent to the LR models obtained by either mapping, it follows from the continuity of the mapping from $\boldsymbol{\theta}$ to $\mathbf{w}$, that the CLL function in the standard BN parameterization also has only global optima (Roos et al., 2005). This implies that all our algorithms should converge to the maximal CLL value.

It is important to pick good initial values for the parameters. The common approach to initialize BN parameters is to use the (generative) ML estimates. It is crucial for all three parameter spaces to avoid zero probabilities, therefore we use Laplace's rule, and add one to each frequency. After the initial values of $\boldsymbol{\theta}$ are obtained, we can derive starting values for $\mathbf{w}$ as well. We simply apply the mappings to obtain the initial values for the LR_Roos and LR parameters. This procedure guarantees that all algorithms have the same initial CLL value.

6 Experiments

For the experiments we used artificially generated data sets and data sets from the UCI Machine Learning Repository (Blake and Merz, 1998). Artificial data were generated from Bayesian Networks, where the parameter values were obtained by sampling each parameter from a Dirichlet distribution with $\boldsymbol{\alpha} = \mathbf{1}$. We generated data from both simple and complex structures in order to evaluate performance under different scenarios. We generated perfect graphs in the following way: for each attribute we select i parents: the class node and $i-1$ previous nodes(whenever possible) according to indexing, where $i \in \{1, 2, 3\}$.

Table 1 lists the UCI data sets used in the experiments, and their properties. To discretize numeric variables, we used the algorithm described in (Fayyad and Irani, 1993). The ? values in the votes data set were treated as a separate value, not as a missing value.

Table 1: Data sets and their properties

	#Samples	#Attr.	#Class	Max attr. dim.
Glass	214	9	6	4
Pima	768	8	2	4
Satimage	4435	36	7	2
Tic-tac-toe	958	9	2	3
Voting	435	16	2	3

The fitted BNC structures for the UCI data were obtained by (1) connecting the class to all attributes, (2) using a greedy generative structure learning algorithm to add successive arcs. Step (2) was not applied for the Satimage data set for which we fitted the Naive Bayes model. All structures happened to be perfect graphs, thus there was no need for adjustment. In figure 2 we show the structure fitted to the Pima Indians data set. The other structures were of similar complexity.

Figure 3 depicts convergence curves for 9 optimization algorithms on the Pima Indians data.

Table 2 depicts statistics for different UCI and artificial data sets. For each data set i we have starting CLL value CLL^i_{ML} and maximal (over all algorithms) CLL value CLL^i_{max}. We define a threshold $t_i = CLL^i_{max} - 5\% * (CLL^i_{max} - CLL^i_{ML})$. For every algorithm we computed the number of iterations

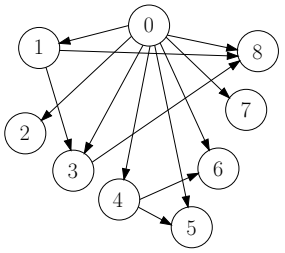

Figure 2: Structure fitted to Pima Indians data. The structure was constructed using a hill climber on the (penalized) generative likelihood.

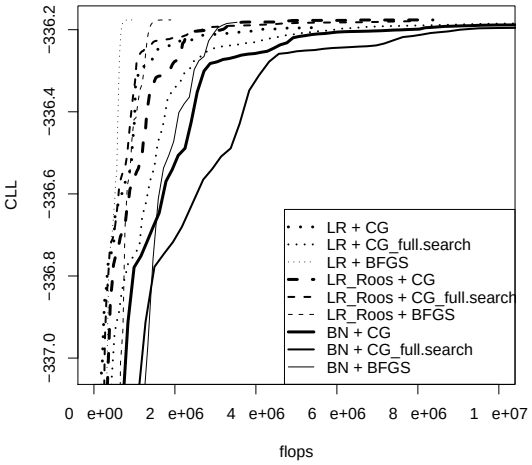

Figure 3: Optimization curves on the Pima Indians data.

(flops) needed to reach the threshold. For each data set these numbers are scaled dividing by the smallest value. The bound of 5% was selected heuristically. This bound is big enough, so all algorithms were able to reach it before 200 iterations and before satisfying the stopping criterion. The bound is small enough, so algorithms make in general quite a few iterations before achieving it. There are some data sets, however, on which all algorithms converge fast to a very high CLL value, and a clear distinction in performance is visible using a very small bound. Pima Indians is an example (5% threshold is set to -337.037). We still see that table 2 contains a quite fair comparison, except that BFGS methods are underestimated for smaller bounds.

It is very interesting to notice that convergence

Data Set	LR			LR_Roos			BN		
	CG	CGB	BFGS	CG	CGB	BFGS	CG	CGB	BFGS
Glass	2.1	1.9	1.0	3.0	2.0	1.1	2.4	1.9	1.0
Pima	1.3	1.5	1.8	1.5	1.2	1.8	1.0	1.0	1.8
Satimage	8.2	2.3	1.0	3.3	2.0	1.1	3.0	1.0	1.3
Tic-tac-toe	3.9	2.2	1.5	2.1	1.1	1.2	2.2	1.0	1.3
Voting	3.2	1.8	1.2	1.9	1.3	1.0	1.5	1.0	1.1
1.5.2-3.100	2.8	1.4	1.2	2.3	1.3	1.1	2.0	1.0	1.2
2.5.2-3.100	2.6	2.1	1.4	1.8	1.6	1.3	1.4	1.0	1.0
3.5.2-3.100	2.7	1.8	1.3	2.2	1.0	1.2	1.3	1.2	1.2
1.5.2-3.1000	2.2	2.2	2.2	1.5	1.5	2.5	1.0	1.0	1.5
2.5.2-3.1000	2.2	2.0	1.8	1.2	1.2	1.8	1.0	1.0	2.5
3.5.2-3.1000	3.6	2.2	1.0	1.8	1.6	1.0	1.3	1.3	1.0
1.30.2-3.1000	2.0	1.2	1.0	1.5	1.2	1.5	1.2	1.0	1.2
2.30.2-3.1000	2.0	1.8	2.2	1.6	1.2	1.4	1.4	1.0	1.2
3.30.2-3.1000	2.2	1.2	2.0	1.6	1.0	1.6	1.4	1.0	1.2
1.5.5.1000	3.1	3.2	1.8	2.3	2.3	1.5	1.1	1.1	1.0
2.5.5.1000	6.7	6.9	2.3	3.1	3.0	1.4	1.5	1.5	1.0
3.5.5.1000	4.8	4.2	1.8	2.3	2.2	1.1	1.7	1.7	1.0
mean	3.3	2.4	1.6	2.1	1.6	1.4	1.6	1.2	1.3

Data Set	LR			LR_Roos			BN		
	CG	CGB	BFGS	CG	CGB	BFGS	CG	CGB	BFGS
Glass	1.0	2.4	3.3	2.6	4.1	8.3	5.9	8.1	12.6
Pima	1.0	2.4	1.6	1.8	2.5	3.1	4.0	6.1	11.0
Satimage	4.6	3.0	1.0	4.6	7.1	3.3	11.5	6.1	7.6
Tic-tac-toe	2.0	3.2	1.0	1.3	1.7	1.3	6.1	4.0	6.0
Voting	1.2	1.9	1.9	1.0	1.8	3.5	3.6	4.2	15.8
1.5.2-3.100	1.0	1.7	1.0	1.2	2.1	1.8	2.9	2.9	5.1
2.5.2-3.100	1.2	2.8	3.5	1.0	2.6	7.4	2.1	3.2	13.2
3.5.2-3.100	1.0	2.4	2.3	1.1	1.4	5.3	2.1	4.4	21.2
1.5.2-3.1000	1.0	2.0	1.7	1.0	1.9	3.9	1.5	2.4	5.5
2.5.2-3.1000	1.5	3.2	2.2	1.0	2.0	4.6	2.3	3.6	25.0
3.5.2-3.1000	1.7	2.6	1.4	1.0	2.2	3.5	2.2	4.3	13.5
1.30.2-3.1000	1.9	3.7	1.0	3.4	8.4	4.3	12.1	17.4	16.6
2.30.2-3.1000	1.0	3.1	2.2	1.2	3.2	3.3	4.9	7.4	11.8
3.30.2-3.1000	1.2	2.3	5.8	1.0	2.4	12.5	4.1	7.6	36.6
1.5.5.1000	1.1	2.8	1.5	1.0	2.4	1.9	1.6	2.4	2.8
2.5.5.1000	2.3	5.6	11.1	1.0	2.2	10.1	1.7	2.4	11.6
3.5.5.1000	2.6	5.4	101.5	1.0	2.3	98.1	2.4	3.8	137.5
mean	1.6	3.0	8.5	1.5	3.0	10.4	4.2	5.3	20.8

Table 2: Convergence speed in terms of iterations (top) and flops (bottom). Artificial data sets are named in the following way: [number of parents of each attribute].[number of attributes].[domain size of each attribute].[sample size]. Domain size denoted as 2-3 is randomly chosen between 2 and 3 with equal probabilities.

with regard to iterations is faster in BN and LR_Roos parameter spaces. It seems that overparameterization is advantageous in this respect. This might be explained by the fact that there is only a single global optimum in the LR case, whereas there are many global optima in case of BN and LR_Roos. Thus, in the latter case one can 'quickly' converge to the closest optimum. Overparameterization, however, is also an additional burden, as witnessed by the flops count; this is especially true for the BFGS method.

We observe that BFGS is generally winning, with CGB being close, in terms of iterations. Considering flops, CGB is definitely losing to CG in all parameterizations. CG seems to be the best technique, though it might be very advantageous to use BFGS for simple (with respect to the number of parents and the domain size of attributes) structures, in combination with our LR mapping.

We have both theoretical and practical evidence that our LR parameterization is the best for relatively simple structures. So the general conclusion is to use LR + BFGS, LR + CG and LR_Roos + CG in order of growing structure complexity. The size of the domain and the number of parents mainly influence our choice. On the basis of flop counts, the BN parameterization is never preferred in our experiments. Finally, we note that our LR parameterization was the best on 4 out of 5 UCI data sets.

7 Conclusion

We have studied the efficiency of discriminative scoring of BNCs using alternative parameterizations of the conditional distribution of the class variable. In case the canonical form of the BNC is a perfect graph, there is a choice between at least three parameterizations. We found out that it is wise to exploit perfectness by optimizing in LR or LR_Roos spaces. Based on the experiments we have performed, we would suggest to use LR + BFGS, LR + CG and LR_Roos + CG in order of growing structure complexity. If only one method is to be selected, we suggest LR_Roos + CG. It works pretty well on any type of data set, plus the mapping is very straightforward, which makes the initialization step easy to implement.

References

J. A. Anderson. 1982. Logistic discrimination. In P. R. Krishnaiah and L. N. Kanal, editors, *Classification, Pattern Recognition and Reduction of Dimensionality*, volume 2 of *Handbook of Statistics*, pages 169–191. North-Holland.

C.L. Blake and C.J. Merz. 1998. UCI repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/mlrepository.html].

U. Fayyad and K. Irani. 1993. Multi-interval discretization of continuous valued attributes for classification learning. In *Proceedings of IJCAI-93 (volume 2)*, pages 1022–1027. Morgan Kaufmann.

R. Greiner, S. Xiaoyuan, B. Shen, and W. Zhou. 2005. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Machine Learning*, 59:297–322.

T. Minka. 2001. Algorithms for maximum-likelihood logistic regression. Technical Report Statistics 758, Carnegie Mellon University.

J. C. Nash. 1990. *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation (2nd ed.)*. Hilger.

F. Pernkopf and J. Bilmes. 2005. Discriminative versus generative parameter and structure learning of Bayesian network classifiers. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 657–664, New York. ACM Press.

W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. 1992. *Numerical Recipes in C: the art of scientific computing*. Cambridge.

T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri. 2005. On discriminative Bayesian network classifiers and logistic regression. *Machine Learning*, 59:267–296.

G. Santafé, J. A. Lozano, and P. Larrañaga. 2005. Discriminative learning of Bayesian network classifiers via the TM algorithm. In L. Godo, editor, *ECSQARU 2005*, volume 3571 of *Lecture Notes in Computer Science*, pages 148–160. Springer.

C. Sutton and A. McCallum. 2006. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press. To appear.

W.N. Venables and B.D. Ripley. 2002. *Modern Applied Statistics with S (fourth edition)*. Springer, New York.