Maximum Margin Bayesian Network Classifiers

Franz Pernkopf, *Member, IEEE*, Michael Wohlmayr, *Student Member, IEEE*,
Sebastian Tschiatschek, *Student Member, IEEE*

Abstract—We present a maximum margin parameter learning algorithm for Bayesian network classifiers using a conjugate gradient (CG) method for optimization. In contrast to previous approaches, we maintain the normalization constraints of the parameters of the Bayesian network during optimization, i.e. the probabilistic interpretation of the model is not lost. This enables to handle missing features in discriminatively optimized Bayesian networks. In experiments, we compare the classification performance of maximum margin parameter learning to conditional likelihood and maximum likelihood learning approaches. Discriminative parameter learning significantly outperforms generative maximum likelihood estimation for naive Bayes and tree augmented naive Bayes structures on all considered data sets. Furthermore, maximizing the margin dominates the conditional likelihood approach in terms of classification performance in most cases. We provide results for a recently proposed maximum margin optimization approach based on convex relaxation [1]. While the classification results are highly similar, our CG-based optimization is computationally up to orders of magnitude faster. Margin-optimized Bayesian network classifiers achieve classification performance comparable to support vector machines (SVMs) using a fewer number of parameters. Moreover, we show that unanticipated missing feature values during classification can be easily processed by discriminatively optimized Bayesian network classifiers, a case where discriminative classifiers usually require mechanisms to complete unknown feature values in the data first.

Index Terms—Bayesian network classifier, discriminative learning, discriminative classifiers, large margin training, missing features, convex relaxation.

1 Introduction

In statistical learning theory, the PAC bound on the expected risk for unseen data depends on the empirical risk on the training data and a measure for the generalization ability of the empirical model which is directly related to the Vapnik-Chervonenkis (VC) dimension [2]. One of the most successful discriminative classifiers, namely the support vector machine (SVM), finds a decision boundary which maximizes the margin between samples of distinct classes resulting in good generalization properties of the classifier. In contrast, conventional discriminative training methods that rely on the conditional likelihood (CL) optimize only the empirical risk, which is suboptimal. Taskar et al. [3] observed that undirected graphical models can be efficiently trained to maximize the margin. More recently, Guo et al. [1] introduced the maximization of the margin to Bayesian networks using convex optimization. Unlike in undirected graphical models, the main difficulty for Bayesian networks is maintaining the normalization constraints of the local conditional probabilities during parameter learning. In [1], these constraints are relaxed to obtain a convex optimization problem. However, conditions on the graph structure are given, ensuring that the class posterior of the relaxed problem is unchanged in case of re-normalization [4], [5]. Unfortunately, classification results for this algorithm have only been demonstrated on small-scale experiments. Since then, different margin-based training algorithms have been proposed for hidden Markov models in [6], [7] and references therein.

Compared to [1], we maximize the margin in Bayesian network classifiers using a different approach. We keep the sum-to-one constraints which maintain the probabilistic interpretation of the network. This has the particular advantage that summing over missing variables is still possible (as we show in this paper). However, we no longer have a convex optimization problem. Convex problems are desirable in many cases as any local optimum is a global optimum. Collobert et al. [8] show that the optimization of non-convex loss functions in SVMs can lead to sparse solutions (lower number of support vectors) and accelerated training performance. They conclude that the sacrosanct popularity of convex approaches should not pre-empt the exploration of alternative techniques, since they may offer computational advantages. Similar observations are reported in [7] and in this article.

In this paper, we introduce maximum margin (MM) parameter learning for Bayesian network classifiers using a conjugate gradient (CG) method [9]. We treat two cases of discriminative parameter learning: both optimization criteria (CL or MM) are optimized using a CG algorithm. CG-based CL learning for Bayesian networks has been introduced in [10]. Recently, we proposed to use the extended Baum-Welch (EBW) algorithm [11] for optimizing the CL of Bayesian network classifiers [12]. In the speech community, the EBW algorithm is well-known for optimizing the CL of hidden Markov models [11], [13]. EBW offers an *EM-like* parameter update.

• *F. Pernkopf, M. Wohlmayr, and S. Tschiatschek are with the Department of Electrical Engineering, Laboratory of Signal Processing and Speech Communication, Graz University of Technology, Austria.*
E-mail: pernkopf@tugraz.at, michael.wohlmayr@tugraz.at, tschiatschek@tugraz.at

This work was supported by the Austrian Science Fund (Project number P22488-N23) and (Project number S10610).

In fact, it is shown in [14] that the EBW algorithm resembles the gradient descent algorithm for discriminatively optimizing Gaussian mixtures using a particular step size choice in the gradient descent method. In [15], we attempted to use EBW for MM parameter optimization of Bayesian network classifiers. We empirically observed similar results as for CG-based optimization, however the EBW requires a rational objective function which can not be guaranteed anymore. Similarly, we introduced maximum margin learning to Gaussian mixture models using the EBW algorithm [16].

In experiments, we compare the classification performance of generative maximum likelihood (ML) and discriminative MM and CL parameter learning approaches. We show that maximizing the margin dominates the conditional likelihood approach with respect to classification performance for most cases. Furthermore, we provide results for maximum margin optimization using convex relaxation [1]. We achieve highly similar classification rates, whereas our CG-based margin optimization is computationally dramatically less costly. All Bayesian network classifiers use either naive Bayes (NB) or generatively and discriminatively optimized[1] tree augmented naive Bayes (TAN) structures. We also provide results for SVMs showing that margin-optimized Bayesian network classifiers are serious competitors – especially in cases where small-sized and probabilistic models are required. Moreover, we show experiments demonstrating the ability of handling missing feature scenarios. We are particularly interested in situations where unanticipated missing feature values arise during classification, i.e. during testing, which can be easily handled by our discriminatively optimized Bayesian network classifiers. Discriminative models usually require mechanisms to first complete unknown feature values in the data – known as data *imputation* – and then applying the standard classification approach to the completed data. We provide results for two imputation techniques, namely (i) mean value imputation, i.e. the missing feature value is replaced with the mean value of the feature over the entire training data set; (ii) k-nearest neighbor (kNN) value imputation, i.e. the mean value (for discretized data the most frequent value) of the k-nearest neighbors is used as surrogate of the missing feature value. kNN feature value imputation is slow and requires the training data to be available during classification.

The paper is organized as follows: In Section 2, we introduce our notation and briefly review Bayesian networks, ML parameter learning as well as NB and TAN structures. In Section 3, we introduce MM parameter learning. Section 4 summarizes a generative and two discriminative structure learning algorithms used in the experiments. In Section 5, we present experimental results for phonetic classification using the TIMIT speech corpus [17], for handwritten digit recognition using the MNIST [18] and USPS data sets, and for a remote sensing application. Furthermore, experiments for missing feature situations are reported in Section 5.1 and 5.2. In Section 5.3, we show results for margin-based Bayesian network parameter optimization using convex relaxation and provide the runtime for each of the maximum margin parameter learning algorithms. Finally, Section 6 concludes the paper.

1. By "discriminative structure learning", we mean that the aim of optimization is to learn the structure of the network by maximizing a cost function that is suitable for reducing classification errors, such as conditional likelihood or classification rate.

2 Bayesian Network Classifiers

A Bayesian network [19] $\mathcal{B} = \langle \mathcal{G}, \Theta \rangle$ is a directed acyclic graph $\mathcal{G} = (\mathbf{Z}, \mathbf{E})$ consisting of a set of nodes $\mathbf{Z}$ and a set of directed edges $\mathbf{E}$ connecting the nodes. This graph represents factorization properties of the distribution of a set of random variables $\mathbf{Z} = \{Z_1, \ldots, Z_{N+1}\}$, where $|Z_j|$ denotes the cardinality of Z_j. The variables in $\mathbf{Z}$ have values denoted by lower case letters $\mathbf{z} = \{z_1, z_2, \ldots, z_{N+1}\}$. We use boldface capital letters, e.g. $\mathbf{Z}$, to denote a set of random variables and correspondingly boldface lower case letters, e.g. $\mathbf{z}$, denote a set of instantiations (values). Without loss of generality, in Bayesian network classifiers the random variable Z_1 represents the class variable $C \in \{1, \ldots, |C|\}$, where $|C|$ is the number of classes and $\mathbf{X}_{1:N} = \{X_1, \ldots, X_N\} = \{Z_2, \ldots, Z_{N+1}\}$ denotes the set of random variables which model the N attributes of the classifier. In a Bayesian network each node is independent of its non-descendants given its parents. Conditional independencies among variables reduce the computational effort for exact inference on such a graph. The set of parameters which quantify the network is represented by Θ. Each node Z_j is represented as a local conditional probability distribution given its parents Z_{Π_j}. We use $\theta^j_{i|h}$ to denote a specific conditional probability table entry (assuming discrete variables); the probability that variable Z_j takes on its i^{th} value assignment given that its parents Z_{Π_j} take their h^{th} (lexicographically ordered) assignment, i.e. $\theta^j_{i|h} = P_\Theta\left(Z_j = i | Z_{\Pi_j} = h\right)$. Hence, h contains the parent configuration assuming that the first element of h denoted as h_1 is the conditioning class and the remaining elements $h \backslash h_1$ are the conditioning parent attribute values. The training data consists of M independent and identically distributed samples $\mathcal{S} = \{\mathbf{z}^m\}_{m=1}^M = \{(c^m, \mathbf{x}^m_{1:N})\}_{m=1}^M$ where $M = |\mathcal{S}|$. The joint probability distribution of a sample $\mathbf{z}^m$ is determined as

$$P_\Theta\left(\mathbf{Z} = \mathbf{z}^m\right) = \prod_{j=1}^{N+1} P_\Theta\left(Z_j = z^m_j | Z_{\Pi_j} = z^m_{\Pi_j}\right) = \prod_{j=1}^{N+1} \prod_{i=1}^{|Z_j|} \prod_h \left(\theta^j_{i|h}\right)^{u^{j,m}_{i|h}}, \quad (1)$$

where we use $u^{j,m}_{i|h}$ to represent the m^{th} sample in binary form, i.e. $u^{j,m}_{i|h} = \mathbb{1}_{\left\{z^m_j = i \text{ and } z^m_{\Pi_j} = h\right\}}$. Symbol $\mathbb{1}_{\{i=j\}}$ denotes the indicator function, i.e. it equals 1 if the Boolean expression $i = j$ is true and 0 otherwise. The class labels are predicted using the maximum a-posteriori (MAP) estimate obtained by Bayes rule, i.e.

$$P_\Theta\left(C = c|\mathbf{X}_{1:N} = \mathbf{x}_{1:N}^m\right) = \frac{P_\Theta\left(C = c, \mathbf{X}_{1:N} = \mathbf{x}_{1:N}^m\right)}{\sum_{c'=1}^{|C|} P_\Theta\left(C = c', \mathbf{X}_{1:N} = \mathbf{x}_{1:N}^m\right)},$$

where the most likely class c^* is determined as $c^* = \arg\max_{c' \in \{1,\ldots,|C|\}} P_\Theta\left(C = c'|\mathbf{X}_{1:N} = \mathbf{x}_{1:N}^m\right)$.

For the sake of brevity, we only notate instantiations of the random variables in the sequel.

2.1 Generative ML Parameter Learning

The log likelihood function of a fixed structure of $\mathcal{B}$ is

$$LL\left(\mathcal{B}|\mathcal{S}\right) = \sum_{m=1}^{M} \sum_{j=1}^{N+1} \sum_{i=1}^{|Z_j|} \sum_{h} u_{i|h}^{j,m} \log\left(\theta_{i|h}^{j}\right).$$

Maximizing $LL\left(\mathcal{B}|\mathcal{S}\right)$ leads to the ML estimate of the parameters

$$\theta_{i|h}^{j} = \frac{\sum_{m=1}^{M} u_{i|h}^{j,m}}{\sum_{m=1}^{M} \sum_{l=1}^{|Z_j|} u_{l|h}^{j,m}},$$

using Lagrange multipliers to constrain the parameters to a valid normalized probability distribution, i.e. $\sum_{i=1}^{|Z_j|} \theta_{i|h}^{j} = 1$.

2.2 Discriminative CL Parameter Learning

Maximizing CL is tightly connected to minimizing the empirical risk. Unfortunately, CL does not decompose as ML does. Consequently, there is no closed-form solution. The conditional log likelihood (CLL) is

$$\begin{aligned} CLL\left(\mathcal{B}|\mathcal{S}\right) &= \log \prod_{m=1}^{M} P_\Theta\left(c^m|\mathbf{x}_{1:N}^m\right) \qquad (2) \\ &= \sum_{m=1}^{M} \left[\log P_\Theta\left(c^m, \mathbf{x}_{1:N}^m\right) - \log \sum_{c=1}^{|C|} P_\Theta\left(c, \mathbf{x}_{1:N}^m\right)\right]. \end{aligned}$$

A conjugate gradient algorithm [10], [20] or the EBW method [12] have been proposed for maximizing $CLL\left(\mathcal{B}|\mathcal{S}\right)$. For the sake of completeness, we shortly sketch the CG algorithm for CL optimization in the Appendix.

2.3 Structures

In this work, we restrict our experiments to NB and TAN structures defined in the next paragraphs. The NB network assumes that all the attributes are conditionally independent given the class label. This means that, given C, any subset of $\mathbf{X}$ is independent of any other disjoint subset of $\mathbf{X}$. As reported in the literature [21], [22], the performance of the NB classifier is surprisingly good even if the conditional independence assumption between attributes is unrealistic or even false in most of the data. Reasons for the utility of the NB classifier range between benefits from the bias/variance tradeoff perspective [21] to structures that are inherently poor from a generative perspective but good from a discriminative perspective [23]. The structure of the naive Bayes classifier represented as a Bayesian network is illustrated in Figure 1(a).

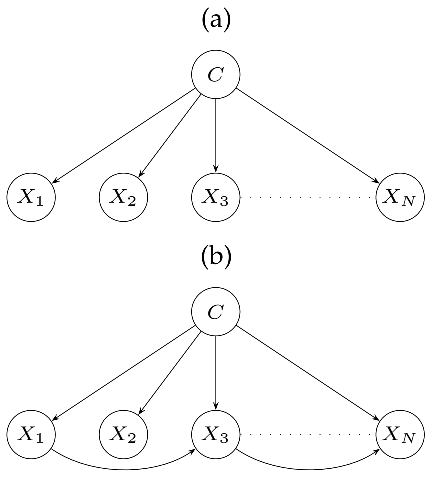


Fig. 1. Bayesian Network: (a) NB, (b) TAN.

In order to overcome some of the limitations of the NB classifier, Friedman et al. [21] introduced the TAN classifier. A TAN is based on structural augmentations of the NB network: Additional edges are added between attributes. Each attribute may have at most one other attribute as an additional parent which means that the tree-width of the attribute induced sub-graph is unity[2], i.e. we have to learn a 1-tree over the attributes. The maximum number of edges added to relax the independence assumption between the attributes is $N - 1$. Thus, two attributes might not be conditionally independent given the class label in a TAN. An example of a TAN network is shown in Figure 1(b).

A TAN network is typically initialized as a NB network and additional edges between attributes are determined through structure learning. Hence, TAN structures are restricted such that the class node remains parent-less, i.e. $C_\Pi = \emptyset$. An extension of the TAN network is to use a k-tree, i.e. each attribute can have a maximum of k attribute nodes as parents. In [20], we noticed that 2-trees over the features do not improve classification performance significantly without regularization. Therefore, we limit the experiments to NB and TAN structures. Many other network topologies have been suggested in the past – a good overview is provided in [26].

2. The tree-width of a graph is defined as the size (i.e. number of variables) of the largest clique of the moralized and triangulated directed graph minus one. Since there are commonly multiple triangulated graphs, the tree-width is defined by the triangulation where the largest clique has the fewest number of variables. More details are given in [24], [25] and references therein.

3 Discriminative margin-based parameter learning

The proposed CG-based maximum margin learning algorithm is developed in the following sections.

3.1 Maximum Margin Objective Function

The multi-class margin [1] of sample m can be expressed as

$$\tilde{d}_{\Theta}^{m} = \min_{c \neq c^m} \frac{P_{\Theta}\left(c^m | \mathbf{x}_{1:N}^m\right)}{P_{\Theta}\left(c | \mathbf{x}_{1:N}^m\right)} = \frac{P_{\Theta}\left(c^m, \mathbf{x}_{1:N}^m\right)}{\max_{c \neq c^m} P_{\Theta}\left(c, \mathbf{x}_{1:N}^m\right)}. \quad (3)$$

Sample m is correctly classified if and only if $\tilde{d}_{\Theta}^{m} > 1$. We replace the maximum operator by the differentiable softmax function $\max_x f(x) \approx \log\left[\sum_x \exp\left(\eta f(x)\right)\right]^{\frac{1}{\eta}}$ parameterized by η, where $\eta \geq 1$ and $f(x)$ is non-negative [6]. In the limit of $\eta \to \infty$ the approximation approaches the maximum operator.[3] Using this we can define the approximate multi-class margin d_{Θ}^{m}. Taking the logarithm we obtain

$$\log d_{\Theta}^{m} = \log P_{\Theta}\left(c^m, \mathbf{x}_{1:N}^m\right) - \frac{1}{\eta} \log \sum_{c \neq c^m} \left(P_{\Theta}\left(c, \mathbf{x}_{1:N}^m\right)\right)^{\eta}. \quad (4)$$

Usually, the maximum margin approach maximizes the margin of the sample with the smallest margin for a separable classification problem [27], i.e. the objective is to maximize $\min_{m=1,\ldots,M} \log d_{\Theta}^{m}$. For a non-separable problem, we aim to relax this by introducing a soft margin, i.e. we focus on samples with $\log d_{\Theta}^{m}$ close to zero. For this purpose, we consider the *hinge* loss function

$$\widetilde{M}\left(\mathcal{B}|\mathcal{S}\right) = \sum_{m=1}^{M} \min\left(1, \lambda \log d_{\Theta}^{m}\right),$$

where the scaling parameter $\lambda > 0$ controls the margin with respect to the loss function and is set by cross-validation. Maximizing this function with respect to the parameters Θ implicitly increases the log-margin, whereas the emphasis is on samples with $\lambda \log d_{\Theta}^{m} < 1$, i.e. samples with a large positive margin have no impact on the optimization. Maximizing $\widetilde{M}\left(\mathcal{B}|\mathcal{S}\right)$ using CG is not straight forward due to the non-differentiability in the derivative at $\lambda \log d_{\Theta}^{m} = 1$. Therefore, we propose to use a *smooth hinge* function $h_{\kappa}(y)$ inspired by the Huber loss [28] which is differentiable in $\mathbb{R}$ and has a similar shape as $\min\left[1, y\right]$:

$$h_{\kappa}(y) = \begin{cases} y + \kappa, & \text{if } y \leq 1 - 2\kappa, \\ 1 - \frac{(y-1)^2}{4\kappa}, & \text{if } 1 - 2\kappa < y < 1, \text{ and} \\ 1, & \text{if } y \geq 1, \end{cases} \quad (5)$$

3. Empirical results showed that the performance of the algorithm is not sensitive to the choice of η for $\eta \geq 5$. The case $\eta = 1$ resembles the classical softmax function which empirically showed a slightly inferior performance.

where κ parameterizes this loss function. For $\kappa \to 0$ the smooth hinge function approaches $\min\left(1, y\right)$. This function requires to divide the data $\mathcal{S}$ into three partitions depending on $y^m = \lambda \log d_{\Theta}^{m}$, i.e. $\mathcal{S}_{\Theta}^{1}$ contains samples where $y^m \leq 1 - 2\kappa$, $\mathcal{S}_{\Theta}^{2}$ consists of samples with a margin in the range $1 - 2\kappa < y^m < 1$, and $\mathcal{S}_{\Theta}^{3} = \mathcal{S} \setminus \left\{\mathcal{S}_{\Theta}^{1} \cup \mathcal{S}_{\Theta}^{2}\right\}$. The smooth hinge function $h_{\kappa}(y)$ parameterized by κ is shown in Figure 2.

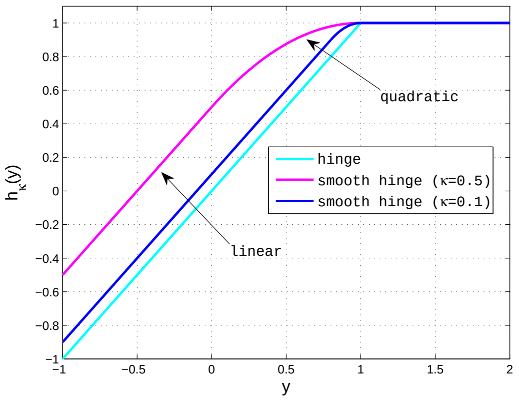


Fig. 2. Differentiable approximation of the hinge loss for $\kappa = 0.5$ and $\kappa = 0.1$.

Similar as in [29] we empirically identified typical values of κ in the range between 0.01 and 0.5. Tuning parameter κ in the given range has a moderate impact on the performance (as we show in experiments). Hence, we suggest to fix this parameter in case of time constraints.

Finally, using the introduced smooth hinge loss our objective function for margin maximization is

$$M\left(\mathcal{B}|\mathcal{S}\right) = \sum_{m \in \mathcal{S}_{\Theta}^{1}} \left(\lambda \log d_{\Theta}^{m} + \kappa\right) + \sum_{m \in \mathcal{S}_{\Theta}^{2}} \left[1 - \frac{\left(\lambda \log d_{\Theta}^{m} - 1\right)^2}{4\kappa}\right] + |\mathcal{S}_{\Theta}^{3}|. \quad (6)$$

This function is differentiable and can be optimized by CG methods.

3.2 CG Algorithm

We use a conjugate gradient algorithm with line-search [30] which requires both the objective function (6) and its derivative. In particular, the *Polak-Ribiere* method is used [9]. The probability $\theta_{i|h}^{j}$ is constrained to $\theta_{i|h}^{j} \geq 0$ and $\sum_{i=1}^{|Z_j|} \theta_{i|h}^{j} = 1$. To incorporate these constraints in the conjugate gradient algorithm we re-parameterize the problem according to

$$\theta_{i|h}^{j} = \frac{\exp\left(\beta_{i|h}^{j}\right)}{\sum_{l=1}^{|Z_j|} \exp\left(\beta_{l|h}^{j}\right)},$$

where $\beta^j_{i|h} \in \mathbb{R}$ is unconstrained. The CG algorithm requires the gradient $\frac{\partial M(\mathcal{B}|\mathcal{S})}{\partial \beta^j_{i|h}}$ which is obtained using the chain rule as

$$\frac{\partial M\left(\mathcal{B}|\mathcal{S}\right)}{\partial \beta^j_{i|h}} = \sum_{k=1}^{|Z_j|} \frac{\partial M\left(\mathcal{B}|\mathcal{S}\right)}{\partial \theta^j_{k|h}} \frac{\partial \theta^j_{k|h}}{\partial \beta^j_{i|h}}. \tag{7}$$

3.3 Derivatives

The derivative of $\frac{\partial M(\mathcal{B}|\mathcal{S})}{\partial \Theta}$ in Eq. (7) is

$$\frac{\partial M\left(\mathcal{B}|\mathcal{S}\right)}{\partial \theta^j_{i|h}} = \sum_{m=1}^{M} s^{m,\lambda}_{\Theta} \frac{\partial \log d^m_{\Theta}}{\partial \theta^j_{i|h}},$$

where $s^{m,\lambda}_{\Theta}$ denotes a sample dependent weight given by

$$s^{m,\lambda}_{\Theta} = \begin{cases} \lambda, & \text{if } m \in \mathcal{S}^1_{\Theta}, \\ -\frac{\lambda}{2\kappa}\left(\lambda \log d^m_{\Theta} - 1\right), & \text{if } m \in \mathcal{S}^2_{\Theta}, \text{ and} \\ 0, & \text{if } m \in \mathcal{S}^3_{\Theta}. \end{cases} \tag{8}$$

When determining the derivative $\log d^m_{\Theta}$ we have to distinguish among two cases: For TAN and NB structures each parameter $\theta^j_{i|h}$ involves the class node value, either $C = i$ for $j = 1$ or $C = h_1$ for $j > 1$ where h_1 denotes the class instantiation $h_1 \in h$. Due to this fact, at most one summand is nonzero when differentiating the term $\sum_{c \neq c^m} \left(P_{\Theta}\left(c, \mathbf{x}^m_{1:N}\right)\right)^{\eta}$ in Eq. (4) with respect to $\theta^j_{i|h}$.

Case A: For the class variable, i.e. $j = 1$ and $h = \emptyset$, the derivative of Eq. (4) after introducing the joint probability of Eq. (1) results in

$$\frac{\partial \log d^m_{\Theta}}{\partial \theta^1_i} = \frac{u^{1,m}_i}{\theta^1_i} - 1\!\!1_{\{i \neq c^m\}} \frac{V^m_i}{\theta^1_i},$$

where we set V^m_i to

$$V^m_i = \frac{\left[P_{\Theta}\left(i, \mathbf{x}^m_{1:N}\right)\right]^{\eta}}{\sum\limits_{c \neq c^m}^{|C|} \left[P_{\Theta}\left(c, \mathbf{x}^m_{1:N}\right)\right]^{\eta}}.$$

Case B: For the attribute variables, i.e. $j > 1$, we differentiate correspondingly and have

$$\frac{\partial \log d^m_{\Theta}}{\partial \theta^j_{i|h}} = \frac{u^{j,m}_{i|h}}{\theta^j_{i|h}} - 1\!\!1_{\{h_1 \neq c^m\}} V^m_{h_1} \frac{v^{j,m}_{i|h\backslash h_1}}{\theta^j_{i|h}},$$

where $v^{j,m}_{i|h\backslash h_1} = 1\!\!1_{\left\{z^m_j = i \text{ and } z^m_{\Pi_j} = h \backslash h_1\right\}}$.
Hence, the gradient $\frac{\partial M(\mathcal{B}|\mathcal{S})}{\partial \Theta}$ for Case A and Case B is

$$\frac{\partial M\left(\mathcal{B}|\mathcal{S}\right)}{\partial \theta^1_i} = \sum_{m=1}^{M} \frac{s^{m,\lambda}_{\Theta}}{\theta^1_i} \left[u^{1,m}_i - 1\!\!1_{\{i \neq c^m\}} V^m_i\right]$$

and

$$\frac{\partial M\left(\mathcal{B}|\mathcal{S}\right)}{\partial \theta^j_{i|h}} = \sum_{m=1}^{M} \frac{s^{m,\lambda}_{\Theta}}{\theta^j_{i|h}} \left[u^{j,m}_{i|h} - 1\!\!1_{\{h_1 \neq c^m\}} V^m_{h_1} v^{j,m}_{i|h\backslash h_1}\right],$$

respectively. These derivatives are further used in Eq. (7) resulting in the required gradient for the CG algorithm. Hence, for Case A we obtain

$$\begin{aligned} \frac{\partial M\left(\mathcal{B}|\mathcal{S}\right)}{\partial \beta^1_i} &= \sum_{m=1}^{M} s^{m,\lambda}_{\Theta} \left[u^{1,m}_i - 1\!\!1_{\{i \neq c^m\}} V^m_i\right] \\ &\quad - \theta^1_i \sum_{m=1}^{M} \sum_{c=1}^{|C|} s^{m,\lambda}_{\Theta} \left[u^{1,m}_c - 1\!\!1_{\{c \neq c^m\}} V^m_c\right], \end{aligned}$$

and for Case B we have

$$\begin{aligned} \frac{\partial M\left(\mathcal{B}|\mathcal{S}\right)}{\partial \beta^j_{i|h}} &= \sum_{m=1}^{M} s^{m,\lambda}_{\Theta} \left[u^{j,m}_{i|h} - 1\!\!1_{\{h_1 \neq c^m\}} V^m_{h_1} v^{j,m}_{i|h\backslash h_1}\right] \\ &\quad - \theta^j_{i|h} \sum_{m=1}^{M} \sum_{l=1}^{|Z_j|} s^{m,\lambda}_{\Theta} \left[u^{j,m}_{l|h} - 1\!\!1_{\{h_1 \neq c^m\}} V^m_{h_1} v^{j,m}_{l|h\backslash h_1}\right]. \end{aligned}$$

4 Structure Learning

This section provides three structure learning heuristics – one generative and two discriminative ones – used in the experiments in Section 5. Note that the parameters during structure learning are optimized generatively using maximum likelihood estimation [19].

4.1 Generative Structure Learning

The conditional mutual information (CMI) [31] between the attributes given the class variable is computed as:

$$I\left(X_i; X_j|C\right) = E_{P(X_i, X_j, C)}\left[\log \frac{P\left(X_i, X_j|C\right)}{P\left(X_i|C\right) P\left(X_j|C\right)}\right],$$

where $E_{P(X)}\left[f(X)\right]$ denotes the expectation of $f(X)$ with respect to $P\left(X\right)$. It measures the information between X_i and X_j in the context of C. In [21], an algorithm for constructing TAN networks using this measure is provided. We briefly review this algorithm in the following:

1) Compute the pairwise CMI $I\left(X_i; X_j|C\right)$ for all $1 \leq i \leq N$ and $i < j \leq N$.
2) Build an undirected 1-tree using the maximal weighted spanning tree algorithm [19] where each edge connecting X_i and X_j is weighted by $I\left(X_i; X_j|C\right)$.
3) Transform the undirected 1-tree into a directed tree. That is, select a root variable and direct all edges away from this root. Add to this tree the class node C and the edges from C to all attributes $X_1, \ldots, X_N$.

This generative structure learning method is abbreviated as CMI in the experiments.

4.2 Greedy Discriminative Structure Learning

This method proceeds as follows: a network is initialized to NB and at each iteration an edge is added that gives the largest improvement of the scoring function, while maintaining a partial 1-tree. Basically, two scoring functions have been considered: the classification rate (CR) [32], [33]

$$CR(\mathcal{B}|\mathcal{S}_V) = \frac{1}{M_V}\sum_{m=1}^{M_V} \mathbb{1}_{\{c^m = \arg\max_{c'} P_\Theta(c'|\mathbf{x}_{1:N}^m)\}}$$

and the CL [34]

$$CL(\mathcal{B}|\mathcal{S}_V) = \prod_{m=1}^{M_V} P_\Theta(c^m|\mathbf{x}_{1:N}^m),$$

where $\mathcal{S}_V = \{(c^m, \mathbf{x}_{1:N}^m)\}_{m=1}^{M_V}$ is the validation data and $M_V = |\mathcal{S}_V|$.

The process of adding edges is terminated when there is no edge which further improves the score. Thus, it might result in a partial 1-tree (forest) over the attributes. This approach is computationally expensive since each time an edge is added, the scores for all $\mathcal{O}(N^2)$ edges need to be re-evaluated due to the discriminative non-decomposable scoring functions we employ. Overall, for learning a k-tree structure, $\mathcal{O}(N^{k+2})$ score evaluations are necessary. In our experiments, we consider the CR score which is directly related to the empirical risk in [2]. The CR is the discriminative criterion that, given sufficient training data, most directly evaluates the objective (small error rate), while an alternative would be to use a convex upper-bound on the 0/1-loss function [35]. Since we are optimizing over a constrained model space (k-trees) regularization is implicit. The CR evaluation can be accelerated by techniques presented in [20], [36]. In the experiments this greedy heuristic is labeled as TAN-CR for 1-tree structures.

Recently, the maximum margin score was introduced for discriminatively optimizing the structure of Bayesian network classifiers [36]. As a search heuristic simulated annealing was used, which offers mechanisms to escape from locally optimal solutions. The maximum margin optimized Bayesian network structures achieve good classification performance.

4.3 Order-based Discriminative Structure Learning

In [37], [20], an order-based greedy algorithm (OMI-CR) has been introduced which is able to find a discriminative TAN structure with only $\mathcal{O}(N^2)$ score evaluations. The order-based algorithm consists of 2 steps:

1) Establishing an ordering: First, a total ordering $\prec$ of the variables $\mathbf{X}_{1:N}$ according to the CMI is established. The feature that is most informative about C is selected first. The next node in the order is the node that is most informative about C conditioned on the first node. More specifically, this step determines an ordered sequence of nodes $\mathbf{X}_{\prec}^{1:N} = \{X_{\prec}^1, X_{\prec}^2, \ldots, X_{\prec}^N\}$ according to

$$X_{\prec}^j \leftarrow \arg\max_{X \in \mathbf{X}_{1:N}\setminus \mathbf{X}_{\prec}^{1:j-1}} \left[I\left(C; X|\mathbf{X}_{\prec}^{1:j-1}\right)\right],$$

where $j \in \{1, \ldots, N\}$.

2) Selecting parents with respect to a given order to form a k-tree: Once the variables are ordered $\mathbf{X}_{\prec}^{1:N}$, the parent $X_{\Pi_j} \in \mathbf{X}_{\Pi_j} = \mathbf{X}_{\prec}^{1:j-1}$ for each $X_{\prec}^j$ ($j \in \{3, \ldots, N\}$) is selected. In case of a small size of $\mathbf{X}_{\Pi_j}$ (i.e. N) and of k a computational costly scoring function to find X_{Π_j} can be used. Basically, either the CL or the CR can be used as cost function to select the parents for learning a discriminative structure. We restrict our experiments to CR for parent selection (empirical results showed a better performance). The parameters are trained using ML learning. A parent is connected to $X_{\prec}^j$ only when CR is improved. Otherwise $X_{\prec}^j$ is left parentless (except C). This might result in a partial 1-tree (forest) over the attributes.

The classification results of the order-based greedy algorithm are not statistical significantly different compared to the greedy algorithm. Similarly, the *SuperParent* algorithm [32] is almost as efficient as OMI-CR achieving slightly lower classification performance [20].

5 EXPERIMENTS

We present results for frame-based phonetic classification using the TIMIT speech corpus [17], for handwritten digit recognition using the MNIST [18] and the USPS data, and for a remote sensing application. In the following, we list the used structure learning algorithms for TAN networks:

- TAN-CMI: Generative TAN structure learning using conditional mutual information (CMI).
- TAN-CR: Discriminative TAN structure learning using the naive greedy heuristic.
- TAN-OMI-CR: Discriminative TAN structure learning using the efficient order-based heuristic.

Once the structure has been determined discriminative parameter learning is performed. This is either done using the proposed CG algorithm to maximize the margin, labeled as CG-MM (see Section 3), or the CL method (see Section 2.2). Additionally, we show results for margin-based Bayesian network optimization using convex relaxation, denoted as CVX-MM, and provide the computational costs for both algorithms.

The parameters are initialized to the ML estimates for all discriminative parameter learning methods.[4] Similar as in [10] we use *cross tuning* to estimate the optimal number of iterations for the CG algorithm to avoid overfitting. Additionally, the value of $\lambda \in [0.001, \ldots, 0.5]$

4. Empirical results showed that the initialization of the Bayesian network to the ML estimates for MM or CL optimization performs better than pure random initialization.

and $\kappa \in [0.01, \ldots, 0.5]$ resulting in the best classification rate is obtained empirically using cross-tuning. We note that instead of early stopping also regularization of the parameters can be used to avoid over-training of the models. In [5], concave priors have been suggested, however, ℓ_1 or ℓ_2-regularization in the unconstrained space of $\beta^j_{i|h}$ is an alternative. In any case, a weight measuring the trade-off between objective function and regularization term has to be determined by cross-validation. So there is no benefit. Empirically, we could not observe any advantage of regularization over early stopping in terms of achieved classification performance.

Continuous features were discretized using recursive minimal entropy partitioning [38] where the quantization intervals were determined using only the training data. Zero probabilities in the conditional probability tables are replaced with small values ε. Further, we used the same data set partitioning for various learning algorithms.

5.1 Handwritten Digit Recognition and Phonetic Classification

5.1.1 Data Characteristics

In the following, we provide details about the used data sets:

TIMIT-4/6 Data: This data set is extracted from the TIMIT speech corpus using the dialect speaking region 4 which consists of 320 utterances from 16 male and 16 female speakers. Speech frames are classified into either four or six classes using 110134 and 121629 samples, respectively. Each sample is represented by 20 mel-frequency cepstral coefficients (MFCCs) and wavelet-based features. We perform classification experiments on data of male speakers (Ma), female speakers (Fe), and both genders (Ma+Fe), all in all resulting in 6 distinct data sets (i.e. Ma, Fe, Ma+Fe $\times$ 4 and 6 classes). The data have been split into 2 mutually exclusive subsets where 70% is used for training and 30% for testing. More details about the features can be found in [39].

MNIST Data: We present results for the handwritten digit MNIST data [18] which contains 60000 samples for training and 10000 digits for testing. We down-sample the gray-level images by a factor of two which results in a resolution of 14×14 pixels, i.e. 196 features.

USPS Data: This data set contains 11000 uniformly distributed handwritten digit images from zip codes of mail envelopes. The data set is split into 8000 images for training and 3000 for testing. Each digit is represented as a 16×16 grayscale image, where again each pixel is considered as feature.

5.1.2 Results

Tables 1, 2, and 3 show the classification rates for MNIST, USPS, and the six TIMIT-4/6 data sets for various learning methods.[5] Additionally, we provide classification performances for SVMs using a radial basis function (RBF) kernel.[6] In particular, for TIMIT-4/6 we only show results for the NB structure. The reason is that the final step of MFCC feature extraction involves a discrete cosine transform, i.e. the features are decorrelated. Hence, we empirically observed that the independence assumptions of the NB structure is a good choice for these data sets.

TABLE 1
Classification results in [%] for MNIST data with standard deviation. Best parameter learning results for each structure are emphasized using bold font.

Classifier	Parameter Learning		
	ML	CG-MM	CG-CL
NB	83.73±0.37	**91.82**±0.27	91.70±0.28
TAN-CMI	91.28±0.28	**94.70**±0.22	93.80±0.24
TAN-OMI-CR	92.01±0.27	**94.94**±0.22	93.39±0.25
TAN-CR	92.58±0.26	**95.12**±0.22	93.94±0.24
SVM ($C^* = 1, \sigma = 0.01$)	**96.40**±0.19		

TABLE 2
Classification results in [%] for USPS data with standard deviation. Best parameter learning results for each structure are emphasized using bold font.

Classifier	Parameter Learning		
	ML	CG-MM	CG-CL
NB	87.10±0.61	**95.23**±0.39	93.67±0.44
TAN-CMI	91.90±0.50	**95.23**±0.39	94.87±0.40
TAN-OMI-CR	92.40±0.48	**95.70**±0.37	94.90±0.40
TAN-CR	92.57±0.48	**96.30**±0.34	95.83±0.36
SVM ($C^* = 1, \sigma = 0.005$)	**97.86**±0.26		

TABLE 3
Classification results in [%] for TIMIT-4/6 data with standard deviation. Best results for each data set are emphasized using bold font.

Data	NB			SVM
	Parameter Learning			$C^* = 1$
	ML	CG-MM	CG-CL	$\sigma = 0.05$
Ma+Fe-4	87.90±0.15	92.09±0.15	92.12±0.16	**92.49**±0.14
Ma-4	88.69±0.25	92.97±0.20	92.81±0.20	**93.30**±0.20
Fe-4	87.67±0.25	91.57±0.21	91.57±0.22	**92.14**±0.21
Ma+Fe-6	81.82±0.20	85.43±0.18	85.41±0.18	**86.24**±0.18
Ma-6	82.26±0.28	86.20±0.26	86.28±0.26	**87.19**±0.25
Fe-6	81.93±0.28	84.85±0.26	85.12±0.26	**86.19**±0.25
Average	84.85	88.67	88.69	**89.38**

The classification rate is improving for more complex structures using ML parameter learning. Discrimina-

5. The average CR over the six TIMIT-4/6 data sets is determined by weighting the CR of each data set with the number of samples in the test set. These values are accumulated and normalized by the total amount of samples in all test sets.

6. The SVM uses two parameters C^* and σ, where C^* is the penalty parameter for the errors in the non-separable case and σ is the variance parameter for the RBF kernel.

tively optimized structures, i.e. TAN-OMI-CR and TAN-CR significantly outperform generatively learned, i.e. TAN-CMI and NB structures. Discriminative parameter learning produces a significantly better classification performance than ML parameter learning on the same classifier structure. This is especially valid for cases where the structure of the underlying model is not optimized for classification [10], i.e. NB and TAN-CMI.

MM parameter optimization outperforms CL learning for most data sets. However, SVMs outperform our discriminative Bayesian network classifiers on all data sets. For TIMIT-4/6 one reason might be that SVMs are applied to the continuous feature domain. In Table 4 we compare the model complexity, i.e. the number of parameters, between SVMs and the best performing Bayesian network classifier. This table reveals that the Bayesian network uses ~ 108, ~ 66, ~ 212, and ~ 259 times fewer parameters than the SVM for MNIST, USPS, Ma+Fe-4, and Ma+Fe-6, respectively. It is a well-known fact that the number of support vectors in classical SVMs increases linearly with the number of training samples [8]. In contrast, the structure of Bayesian network classifiers naturally limits the number of parameters. A substantial difference is that SVMs determine the number of support vectors automatically while in the case of Bayesian networks the number of parameters is given by the cardinality of the variables and the structure. In this way, the model complexity can be easily controlled by constraints on the structure. We use cross-tuning to select C^* and σ for SVMs and parameter λ, κ, and the number of CG iterations for MM learning of Bayesian networks.

In contrast to SVMs, the used Bayesian network structures are probabilistic generative models – even when discriminatively learned. They might be preferred since it is easy to work with missing features, domain knowledge can be directly incorporated into the graph structure, and it is easy to work with structured data. In this paragraph, we demonstrate that a discriminatively optimized generative model still offers its advantages in the missing feature case. Our MM parameter learning keeps the sum-to-one constraint of the probability distributions. Therefore, we suggest, similarly to the generatively optimized models, to sum over the missing feature values. The interpretation of *marginalizing* over missing features is delicate since the discriminatively optimized parameters might not have anything in common with *consistently* estimated probabilities (such as e.g. maximum likelihood estimation). However, at least empirically there is a strong support for using the density $P(C, \mathbf{X}') = \sum_{\mathbf{X}_{1:N} \setminus \mathbf{X}'} P(C, \mathbf{X}_{1:N})$ where $\mathbf{X}'$ is a subset of the features $\mathbf{X}_{1:N}$. This computation is tractable if the complexity class of $P(C, \mathbf{X}_{1:N})$ is limited (e.g. 1-tree) and the variable order in the summation is chosen appropriately. In contrast, classical discriminative models are inherently conditional and it is not possible to obtain $p(C|\mathbf{X}')$ from $p(C|\mathbf{X}_{1:N})$. In particular, this holds for SVMs, logistic regression, and multi-layered perceptrons. These models commonly require *imputation* techniques to first complete missing feature values in the data. Then the classification approach is applied on the completed data.

We are particularly interested in the case where arbitrary sets of missing features for each classification sample can occur during testing.[7] In such a case, it is not possible to re-train the model for each potential set of missing features without also memorizing the training set. In Figure 3(a), we present the classification performance of discriminative and generative structures using ML parameter learning on the MNIST data assuming missing features. The x-axis denotes the number of missing features. The curves are the average over 100 classifications of the test data with uniformly at random selected missing features. We use exactly the same missing features for each classifier. We observe that discriminatively structured Bayesian network classifiers outperform TAN-CMI-ML even in the case of missing features. This demonstrates, at least empirically, that discriminatively structured generative models do not lose their ability to impute missing features.

In Figure 3(b), we show for the same data set and experimental setup that the classification performance of a discriminatively parameterized NB classifier may be superior to a generatively parameterized NB model in the case of missing features. In particular, this advantage holds for up to $\sim$80 missing features. For a larger number of missing features the performance of NB-ML is more robust. Additionally, NB-CG-MM seems to be more robust to increasing number of missing features compared to NB-CG-CL. This can be attributed to the better generalization property of a margin-optimized classifier.

5.2 Remote Sensing

We use a hyperspectral remote sensing image of the Washington D.C. Mall area containing 191 spectral bands having a spectral width of 5-10 nm.[8] As ground reference a classification performed at Purdue University was used containing 7 classes, namely roof, road, grass, trees, trail, water, and shadow.[9] The aerial image using bands 63, 52, and 36 for red, green, and blue colors, respectively, and the reference image are shown in Figure 4(a) and (b). The image contains 1280×307 hyperspectral pixels, i.e. 392960 samples. We arbitrarily choose 5000 samples of each class to learn the classifier. This remote sensing application is in particular interesting for our classifiers since spectral bands might be missing or should be neglected due to atmospheric effects. For example radiation within the visible range should be neglected in case of clouds or darkness.

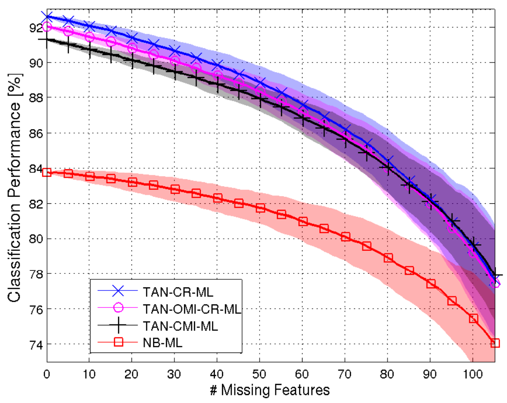
7. Note that we do not consider missing features during training of the classifiers.

8. http://cobweb.ecn.purdue.edu/˜biehl/MultiSpec/hyperspectral.html

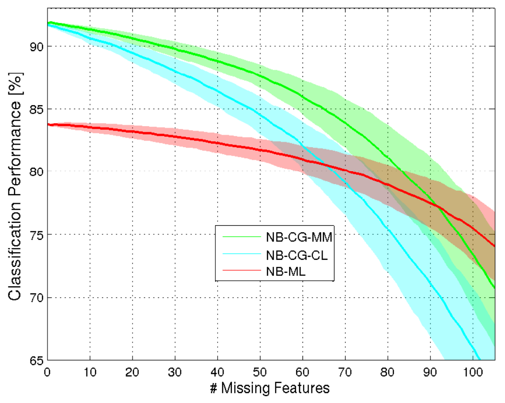
9. http://cobweb.ecn.purdue.edu/˜landgreb/Hyperspectral.Ex.html

TABLE 4
Model complexity for best Bayesian network (BN) and SVM.

Data	N	Number of SVs	Number of SVM parameters	Number of BN parameters
MNIST	196	17201	3371396	31149
USPS	256	3837	982272	14689
TIMIT-4/6 (Ma+Fe-4)	20	13146	262920	1239
TIMIT-4/6 (Ma+Fe-6)	20	24350	487000	1877
Washington D.C. Mall	191	11934	2279394	62566

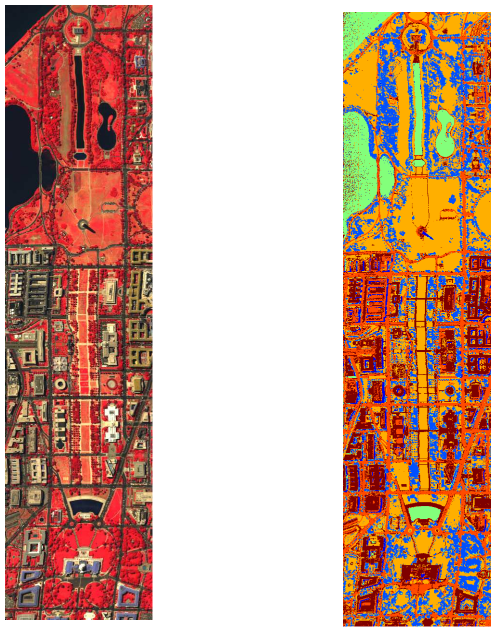


(a)



(b)

Fig. 3. Classification performance on MNIST assuming missing features. The x-axis denotes the number of missing features and the shaded regions correspond to the standard deviation over 100 classifications: (a) Different structure learning methods with generative parameterization; (b) Different discriminative parameter learning methods on NB structure.



(a) (b)

Fig. 4. Washington D.C. Mall: (a) Pseudo color image of spectral bands 63, 52, and 36; (b) Reference image.

TABLE 5
Classification results in $[\%]$ for Washington D.C. Mall data with standard deviation. Best parameter learning result is emphasized using bold font.

NB			SVM
Parameter Learning			$C^* = 1$
ML	CG-MM	CG-CL	$\sigma = 0.05$
81.07 ±0.06	**89.34** ±0.05	87.01 ±0.05	88.98 ±0.05

We use various introduced generative and discriminative parameter learning algorithms on the NB network structure. The classification performances are shown in Table 5.

Remarkably, NB-CG-MM slightly outperforms SVMs in this experiment. Additionally, the Bayesian network employs ~ 36 times fewer parameters than the SVM (see Table 4). Figure 5 shows the influence of parameter κ in the loss function (6) for $\lambda = 0.02$ on the classification performance. The classification rate slightly improves for $\kappa = 0.5$. However, the impact is moderate. Note that the selection of κ is based on the cross-validation performance on the training data.

Similar as for MNIST in Section 5.1, we report classification results for NB-ML, NB-CG-MM, and NB-CG-CL assuming at random missing features during classification in Figure 6. The x-axis denotes the number of missing features. We average the performances over 100 classifications of the test data with randomly missing features. The standard deviation indicates that the resulting differences are significant for a moderate number of missing features. Discriminatively parameterized

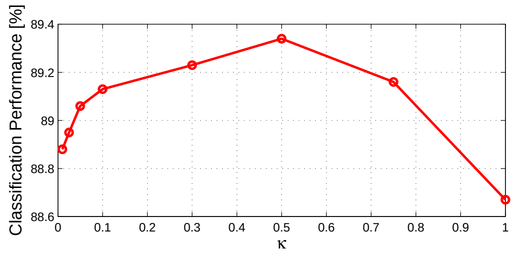


Fig. 5. Influence of parameter κ of the loss function on the classification rate for $\lambda = 0.02$.

NB classifiers outperform NB-ML in the case of up to 150 missing features. Furthermore, we present results for SVMs where first *imputation* methods are used to complete missing feature values in the data. Afterwards, SVMs are applied on the completed data. In particular, we use two imputation approaches: (i) mean value imputation (the missing value is replaced with the mean value of the feature of the training data set); (ii) kNN value imputation – the missing value is replaced with the mean value (for discretized data the most frequent value) of the k-nearest neighbors. The neighbors of a sample with missing features are determined by the Euclidean distance in the relevant subspace. In the special case where k equals the number of training instances M this method is identical to mean value imputation. We use $k = 5$. As shown in Figure 6, mean value imputation degrades the classification performance of SVMs in case of missing features significantly. Handling missing features with NB classifiers is easy since we can simply neglect the conditional probability of the missing feature Z_j in Eq. (1), i.e. the joint probability is the product of the available features only.

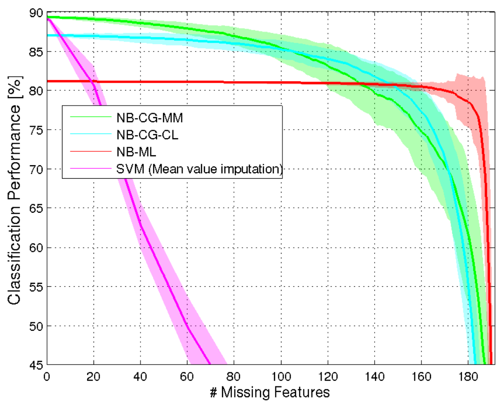


Fig. 6. Washington D.C. Mall: Classification results for NB-ML, NB-CG-MM, NB-CG-CL, and SVMs (using mean value imputation) assuming missing features.

Figure 7 shows kNN value imputation results for SVMs and NB-CG-MM. kNN feature value imputation is slow and requires the training data during classification of samples with at random missing features. However, it provides more information to the classifier compared to simple summation over the missing feature values as shown for the NB-CG-MM case.

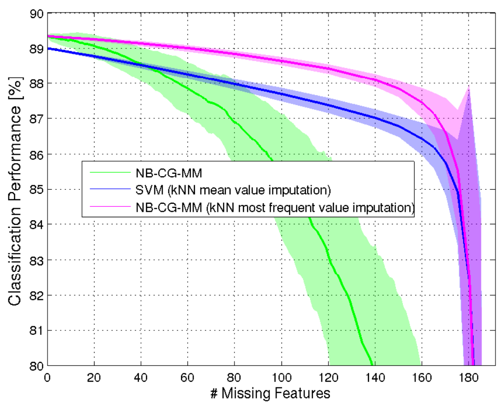


Fig. 7. Washington D.C. Mall: Classification results for NB-CG-MM (summation over missing feature values), NB-CG-MM (using kNN most frequent value imputation), and SVMs (using kNN mean value imputation) assuming missing features.

5.3 Margin Optimization using Convex Relaxation

In this section, we compare our CG-based margin optimization to a recently proposed approach using convex relaxation [1] in terms of classification accuracy and computational efficiency. First we provide a short introduction to convex relaxation for margin maximization and give details on solving the convex problem for our data. Unfortunately, Guo et al. [1] only provided results on small-scale experiments, i.e. 50 samples and up to 36 features.

5.3.1 *Background*

Guo et al. [1] proposed to solve the maximum margin parameter learning problem for Bayesian network classifiers by reformulating it as a convex optimization problem. They introduced the parameter vector $\mathbf{w}$ with elements $w^j_{i|h} = \log(\theta^j_{i|h})$ (in some order) and, using the same order for the elements, the feature vectors $\phi(\mathbf{z}^m)$ with elements $u^{j,m}_{i|h}$. Then, the probability of sample $\mathbf{z}^m$ can be written as $P_\Theta(\mathbf{Z} = \mathbf{z}^m) = \exp(\phi(\mathbf{z}^m)^T\mathbf{w})$, where $\phi(\mathbf{z}^m)^T$ denotes the transpose of $\phi(\mathbf{z}^m)$. The logarithm of the multi-class margin (3) of the m-th sample becomes

$$\log d^m_\Theta = \min_{c \neq c^m} \left[\phi(c^m, \mathbf{x}^m_{1:N}) - \phi(c, \mathbf{x}^m_{1:N})\right]^T \mathbf{w}.$$

In this way, the problem of learning the maximum margin parameters of the Bayesian network can be recast as

$$\underset{\gamma,\mathbf{w}}{\text{maximize}}\ \gamma \quad \text{s.t. } \mathbf{\Delta}_{m,c}\mathbf{w} \geq \gamma, \quad \forall m \text{ and } c \neq c^m,$$
$$\gamma \geq 0, \quad \sum_{i=1}^{|Z_j|} \exp(w^j_{i|h}) = 1, \quad \forall j, h,$$

where γ is the logarithm of the minimum of all sample margins and $\mathbf{\Delta}_{m,c} = [\phi(c^m, \mathbf{x}^m_{1:N}) - \phi(c, \mathbf{x}^m_{1:N})]^T$. The first constraint ensures that all sample margins are greater than γ and the third constraint that $\mathbf{w}$ parameterizes a valid Bayesian network, i.e. $\mathbf{w}$ describes valid probability distributions.

Finally, by introducing one slack variable ϵ_m for each sample $\mathbf{z}^m$, relaxing the constraints on the parameter vector $\mathbf{w}$ and rewriting the objective function, Guo et al. derived the optimization problem

$$\begin{aligned} &\underset{\gamma,\mathbf{w},\epsilon_1,\ldots,\epsilon_M}{\text{minimize}} \ \frac{1}{2\gamma^2} + B\sum_{m=1}^{M}\epsilon_m \qquad (9)\\ &\text{s.t. } \mathbf{\Delta}_{m,c}\mathbf{w} \geq \gamma - \epsilon_m, \quad \forall m \text{ and } c \neq c^m,\\ &\gamma \geq 0, \quad \sum_{i=1}^{|Z_j|}\exp(w^j_{i|h}) \leq 1, \quad \forall j,h,\\ &\epsilon_m \geq 0, \quad \forall m, \end{aligned}$$

for determining the maximum margin parameters. The parameter B can be used to control the slack effect (similar as parameter C in SVMs). The above problem is convex with convex inequality constraints. Hence, any local minimum is also a global minimum. Furthermore, under certain conditions on the structure of the Bayesian network, the (typically) subnormalized parameter vector $\mathbf{w}$ of a solution allows for re-normalization without changing the decision function $P(c|\mathbf{x}_{1:N})$ (see [4], [5] for details).

There are many possibilities to solve the optimization problem in Eq. (9). Any minimization method allowing for a nonlinear objective function and nonlinear convex inequality constraints can be used in principle. We decided to use the large scale solver IPOPT [40] which shows good performance in several applications (see e.g. [41]).[10] IPOPT applies an interior-point method [43] to solve the problem in (9). It requires the objective function, its gradient, the constraint functions, the Jacobian of the constraint functions, and the second derivative of the Lagrangian function. To ensure short runtimes and good results we used an adaptive strategy for the barrier parameter, and let the algorithm run for up to 100 iterations or until sufficient precision was achieved.[11] We refer to solutions obtained by IPOPT as CVX-MM.

5.3.2 Experimental Comparison

Table 6 and Table 7 show the classification rates and runtimes for the different algorithms and datasets, respectively. The classification rates of CVX-MM are slightly better than those of CG-MM, while the proposed algorithm CG-MM[12] is up to orders of magnitude faster.

For USPS the training data is separable with large margin by a NB classifier, i.e. there exists a probability distribution that factors according to a NB network for which all samples in the training set are classified correctly and for which samples from different classes are separated by a large margin. Therefore, an optimal solution of (9) has small objective for a large range of the parameter B. This complicates the choice of B as well as the tolerance settings for the interior-point optimizer (the optimization problem has to be solved with high precision while keeping the optimization tractable). In our experiments we were not able to find a setting such that the achieved classification rate on the test set is larger than 90.90% which is much smaller than the classification rate of CG-MM.

The large computational requirements of CVX-MM are caused by the convex formulation in Eq. (9): there is one inequality for each conditional probability of the network and for every additional training sample the number of inequalities increases by $|C|$, i.e. the number of classes. Further, there is an additional slack variable resulting in an increase of the dimension of the search space. The used test sets were the same as described above. The runtime experiments were performed on a personal computer with 2.8 GHz CPUs, 16 GB of memory, not exploiting any (multicore) parallelization. Furthermore, we fixed the regularization parameter B to 1 for the MNIST data because of time reasons, but selected it using cross tuning for the TIMIT-4/6 and USPS data.

TABLE 6
Classification rate (CR) in [%] for different data using a naive Bayes classifier.

	Parameter Learning			
	ML	CG-MM	CVX-MM	
Data	CR	CR	CR	B
MNIST	83.73	91.82	92.04	1
USPS	87.10	95.23	90.90	1
Ma+Fe-4	87.90	92.09	92.31	$3.9 \cdot 10^{-3}$
Ma-4	88.69	92.97	93.09	$3.9 \cdot 10^{-3}$
Fe-4	87.67	91.57	91.82	$3.9 \cdot 10^{-3}$
Ma+Fe-6	81.82	85.43	85.61	$3.9 \cdot 10^{-3}$
Ma-6	82.26	86.20	86.67	$3.9 \cdot 10^{-3}$
Fe-6	81.93	84.85	85.46	$3.9 \cdot 10^{-3}$

Convex relaxation for margin optimization is interesting due to its sound theoretical background. However, without further algorithmic developments its practical application seems to be limited to applications using only few training data and a low number of features. In contrast, the proposed method for maximum margin parameter learning can deal with large sets of training data efficiently and achieves comparable classification rates. Furthermore, we observed superior runtime per-

10. We used IPOPT 3.9.2 in conjunction with MUMPS 4.9.2 [42], a parallel sparse direct solver. IPOPT was compiled with Lapack 3.2.1 and BLAS from the Netlib repository (version from March 2007). IPOPT is typically faster than the function *fmincon* of MATLAB for this type of optimization problem.

11. Good classifiers do not require highly accurate solutions. Hence, the tolerance for the objective function is set to 10^{-1} – reducing the runtime of the algorithm compared to using default tolerance settings.

12. CG-MM is implemented in MATLAB.

TABLE 7
Runtimes in $[s]$ for different data using a naive Bayes classifier (B as in Table 6).

	Parameter Learning	
Data	CG-MM	CVX-MM
MNIST	833	54 hours
USPS	113	21 hours
Ma+Fe-4	391	1338
Ma-4	168	842
Fe-4	87	844
Ma+Fe-6	202	4566
Ma-6	241	3505
Fe-6	108	3002

formance of the proposed method in the conducted experiments.

6 Conclusion

We derived a discriminative parameter learning algorithm for Bayesian network classifiers based on maximizing the margin. For margin optimization we introduced a conjugate gradient algorithm. In contrast to previous work on margin optimization in probabilistic models, we kept the sum-to-one constraint which maintains the probabilistic interpretation of the network, e.g. summation over missing variables is still possible. In the experiments, we treat two cases of discriminative parameter learning – both optimization criteria (CL or MM) were optimized with the CG method. Furthermore, we applied various parameter learning algorithms on naive Bayes and generatively and discriminatively optimized TAN structures. Discriminative parameter learning significantly outperforms ML parameter estimation. Furthermore, maximizing the margin slightly improves the classification performance compared to CL parameter optimization in most cases.

Additionally, we provided empirical results for a maximum margin optimization approach based on convex relaxation. The classification results of both maximum margin parameter learning approaches are almost identical, whereas the computational requirements of our CG-based optimization are up to orders of magnitude lower. Margin-optimized Bayesian networks perform on par with SVMs in terms of classification rate, however the Bayesian network classifiers require fewer parameters than the SVM and can directly deal with missing features, a case where discriminative classifiers usually require imputation techniques.

Acknowledgments

The authors thank the anonymous reviewers for useful comments that improved the quality of the paper. Thanks to Jeff Bilmes for discussions and support in writing this paper.

Appendix: CL Parameter Learning

The CG algorithm relies on the gradient of the objective function given as

$$\frac{\partial CLL(\mathcal{B}|\mathcal{S})}{\partial \theta^j_{i|h}} = \sum_{m=1}^{M}\left[\frac{\partial}{\partial \theta^j_{i|h}} \log P_\Theta\left(c^m, \mathbf{x}^m_{1:N}\right) - \frac{\sum_{c=1}^{|C|}\frac{\partial}{\partial \theta^j_{i|h}} P_\Theta\left(c, \mathbf{x}^m_{1:N}\right)}{\sum_{c=1}^{|C|} P_\Theta\left(c, \mathbf{x}^m_{1:N}\right)}\right].$$

Similar as in Section 3.3, we distinguish two cases for differentiating $CLL(\mathcal{B}|\mathcal{S})$, i.e. either $C = i$ for $j = 1$ (Case 1) or $C = h_1$ for $j > 1$ (Case 2).

Case 1: For the class variable, i.e. $j = 1$ and $h = \emptyset$, we get

$$\frac{\partial CLL(\mathcal{B}|\mathcal{S})}{\partial \theta^1_i} = \sum_{m=1}^{M}\left[\frac{u^{1,m}_i}{\theta^1_i} - \frac{W^m_i}{\theta^1_i}\right],$$

using Eq. (1) for differentiating the first term (omitting the sum over j and h) and we introduced the class posterior $W^m_i = P_\Theta\left(i|\mathbf{x}^m_{1:N}\right)$ as

$$W^m_i = \frac{P_\Theta\left(i, \mathbf{x}^m_{1:N}\right)}{\sum_{c=1}^{|C|} P_\Theta\left(c, \mathbf{x}^m_{1:N}\right)}.$$

Case 2: For the attribute variables, i.e. $j > 1$, we differentiate correspondingly and have

$$\frac{\partial CLL(\mathcal{B}|\mathcal{S})}{\partial \theta^j_{i|h}} = \sum_{m=1}^{M}\left[\frac{u^{j,m}_{i|h}}{\theta^j_{i|h}} - W^m_{h_1}\frac{v^{j,m}_{i|h\backslash h_1}}{\theta^j_{i|h}}\right],$$

where $W^m_{h_1} = P_\Theta\left(h_1|\mathbf{x}^m_{1:N}\right)$ is the posterior for class h_1 and sample m and $v^{j,m}_{i|h\backslash h_1} = \mathbb{1}_{\left\{z^m_j = i \text{ and } z^m_{\Pi_j} = h\backslash h_1\right\}}$.

The conditional log likelihood given in Eq. (2) can be optimized by a conjugate gradient algorithm using line-search in a similar manner as given in Section 3.2. Again, we re-parameterize the problem to incorporate the constraints on $\theta^j_{i|h}$ in the conjugate gradient algorithm. This requires the gradient of $CLL(\mathcal{B}|\mathcal{S})$ with respect to $\beta^j_{i|h}$ which is computed using the chain rule as

$$\begin{aligned}\frac{\partial CLL(\mathcal{B}|\mathcal{S})}{\partial \beta^1_i} &= \sum_{k=1}^{|Z_j|}\frac{\partial CLL(\mathcal{B}|\mathcal{S})}{\partial \theta^1_k}\frac{\partial \theta^1_k}{\partial \beta^1_i}\\ &= \sum_{m=1}^{M}\left[u^{1,m}_i - W^m_i\right] - \theta^1_i\sum_{m=1}^{M}\sum_{c=1}^{|C|}\left[u^{1,m}_c - W^m_c\right]\end{aligned}$$

for Case 1. Similarly for Case 2, the gradient is

$$\frac{\partial CLL(\mathcal{B}|\mathcal{S})}{\partial \beta^j_{i|h}} = \sum_{m=1}^{M}\left[u^{j,m}_{i|h} - W^m_{h_1}v^{j,m}_{i|h\backslash h_1}\right] - \theta^j_{i|h}\sum_{m=1}^{M}\sum_{l=1}^{|Z_j|}\left[u^{j,m}_{l|h} - W^m_{h_1}v^{j,m}_{l|h\backslash h_1}\right].$$

References

[1] Y. Guo, D. Wilkinson, and D. Schuurmans, "Maximum margin Bayesian networks," in *International Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005, pp. 233–242.

[2] V. Vapnik, *Statistical learning theory*. Wiley & Sons, 1998.

[3] B. Taskar, C. Guestrin, and D. Koller, "Max-margin Markov networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[4] H. Wettig, P. Grünwald, T. Roos, P. Myllymäki, and H. Tirri, "When discriminative learning of bayesian network parameters is easy," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003, pp. 491 – 496.

[5] T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri, "On discriminative Bayesian network classifiers and logistic regression," *Machine Learning*, vol. 59, pp. 267–296, 2005.

[6] F. Sha and L. Saul, "Comparison of large margin training to other discriminative methods for phonetic recognition by hidden Markov models," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2007, pp. 313–316.

[7] G. Heigold, T. Deselaers, R. Schlüter, and H. Ney, "Modified MMI/MPE: A direct evaluation of the margin in speech recognition," in *International Conference on Machine Learning (ICML)*, 2008, pp. 384–391.

[8] R. Collobert, F. Siz, J. Weston, and L. Bottou, "Trading convexity for scalability," in *International Conference on Machine Learning (ICML)*, 2006, pp. 201–208.

[9] C. Bishop, *Neural networks for pattern recognition*. Oxford University Press, 1995.

[10] R. Greiner, X. Su, S. Shen, and W. Zhou, "Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers," *Machine Learning*, vol. 59, pp. 297–322, 2005.

[11] O. Gopalakrishnan, D. Kanevsky, A. Nàdas, and D. Nahamoo, "An inequality for rational functions with applications to some statistical estimation problems," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 107–113, 1991.

[12] F. Pernkopf and M. Wohlmayr, "On discriminative parameter learning of Bayesian network classifiers," in *European Conference on Machine Learning (ECML)*, 2009, pp. 221–237.

[13] P. Woodland and D. Povey, "Large scale discriminative training of hidden Markov models for speech recognition," *Computer Speech and Language*, vol. 16, pp. 25–47, 2002.

[14] R. Schlüter, W. Macherey, M. B., and H. Ney, "Comparison of discriminative training criteria and optimization methods for speech recognition," *Speech Communication*, vol. 34, pp. 287–310, 2001.

[15] F. Pernkopf and M. Wohlmayr, "Maximum margin Bayesian network classifiers," Institute of Signal Processing and Speech Communication, Graz University of Technology, Tech. Rep., 2010.

[16] ——, "Large margin learning of Bayesian classifiers based on gaussian mixture models," in *European Conference on Machine Learning (ECML)*, 2010, pp. 50–66.

[17] L. Lamel, R. Kassel, and S. Seneff, "Speech database development: Design and analysis of the acoustic-phonetic corpus," in *DARPA Speech Recognition Workshop, Report No. SAIC-86/1546*, 1986.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings fo the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[19] J. Pearl, *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, 1988.

[20] F. Pernkopf and J. Bilmes, "Efficient heuristics for discriminative structure learning of Bayesian network classifiers," *Journal of Machine Learning Research*, vol. 11, pp. 2323–2360, 2010.

[21] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, pp. 131–163, 1997.

[22] P. Domingos and M. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, no. 2-3, pp. 103–130, 1997.

[23] J. Bilmes, "Dynamic Bayesian multinets," in *16th Inter. Conf. of Uncertainty in Artificial Intelligence (UAI)*, 2000, pp. 38–45.

[24] R. Cowell, A. Dawid, S. Lauritzen, and D. Spiegelhalter, *Probabilistic networks and expert systems*. Springer Verlag, 1999.

[25] C. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[26] S. Acid, L. de Campos, and J. Castellano, "Learning Bayesian network classifiers: Searching in a space of partially directed acyclic graphs," *Machine Learning*, vol. 59, pp. 213–235, 2005.

[27] B. Schölkopf and A. Smola, *Learning with kernels: Support Vector Machines, regularization, optimization, and beyond*. MIT Press, 2001.

[28] P. Huber, "Robust estimation of a location parameter," *Annals of Statistics*, vol. 53, pp. 73–101, 1964.

[29] O. Chapelle, "Training a support vector machine in the primal," *Neural Computation*, vol. 19, no. 5, pp. 1155–1178, 2007.

[30] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical recipes in C*. Cambridge Univ. Press, 1992.

[31] T. Cover and J. Thomas, *Elements of information theory*. John Wiley & Sons, 1991.

[32] E. Keogh and M. Pazzani, "Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches," in *Workshop on Artificial Intelligence and Statistics*, 1999, pp. 225–230.

[33] F. Pernkopf, "Bayesian network classifiers versus selective k-NN classifier," *Pattern Recognition*, vol. 38, no. 3, pp. 1–10, 2005.

[34] D. Grossman and P. Domingos, "Learning Bayesian network classifiers by maximizing conditional likelihood," in *Inter. Conf. of Machine Lerning (ICML)*, 2004, pp. 361–368.

[35] P. Bartlett, M. Jordan, and J. McAuliffe, "Convexity, classification, and risk bounds," *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 138–156, 2006.

[36] F. Pernkopf and M. Wohlmayr, "Stochastic margin-based structure learning of Bayesian network classifiers," Laboratory of Signal Processing and Speech Communication, Graz University of Technology, Tech. Rep., 2011.

[37] F. Pernkopf and J. Bilmes, "Order-based discriminative structure learning for Bayesian network classifiers," in *International Symposium on Artificial Intelligence and Mathematics*, 2008.

[38] U. Fayyad and K. Irani, "Multi-interval discretizaton of continuous-valued attributes for classification learning," in *Joint Conf. on Artificial Intelligence*, 1993, pp. 1022–1027.

[39] F. Pernkopf, T. Van Pham, and J. Bilmes, "Broad phonetic classification using discriminative Bayesian networks," *Speech Communication*, vol. 143, no. 1, pp. 123–138, 2008.

[40] A. Wächter and L. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, 2006.

[41] L. Biegler and V. Zavala, "Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization," *Computers & Chemical Engineering*, vol. 33, no. 3, pp. 575–582, 2009.

[42] P. Amestoy, I. Duff, J.-Y. L'Excellent, and J. Koster, "MUMPS: A general purpose distributed memory sparse solver," in *5th International Workshop on Applied Parallel Computing*. Springer-Verlag, 2000, pp. 122–131.

[43] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, March 2004.

Franz Pernkopf received his MSc (Dipl. Ing.) degree in Electrical Engineering at Graz University of Technology, Austria, in summer 1999. He earned a PhD degree from the University of Leoben, Austria, in 2002. In 2002 he was awarded the Erwin Schrödinger Fellowship. He was a Research Associate in the Department of Electrical Engineering at the University of Washington, Seattle, from 2004 to 2006. Currently, he is Associate Professor at the Laboratory of Signal Processing and Speech Communication, Graz University of Technology, Austria. His research interests include machine learning, discriminative learning, graphical models, feature selection, finite mixture models, and image- and speech processing applications.

Michael Wohlmayr graduated from Graz University of Technology in June 2007. He conducted his Master thesis in collaboration with University of Crete, Greece. Since October 2007, he is pursuing the PhD degree at the Signal Processing and Speech Communication Laboratory at Graz University of Technology. His research interests include Bayesian networks, speech and audio analysis, as well as statistical pattern recognition.

Sebastian Tschiatschek received the BSc degree and MSc degree (with distinction) in Electrical Engineering at Graz University of Technology (TUG) in 2007 and 2010, respectively. He conducted his Master thesis during a one-year stay at ETH Zürich, Switzerland. Currently, he is with the Signal Processing and Speech Communication Laboratory at TUG where he is pursuing the PhD degree. His research interests include Bayesian networks, information theory in conjunction with graphical models and statistical pattern recognition.