CONVEX COMBINATIONS OF MAXIMUM MARGIN BAYESIAN NETWORK CLASSIFIERS

Sebastian Tschiatschek and Franz Pernkopf
Department of Electrical Engineering, Laboratory of Signal Processing and Speech Communication
Graz University of Technology, Graz, Austria

Keywords: Bayesian network classifiers, Discriminative learning, Convex relaxation, Maximum margin Bayesian networks, Classifier enhancement, Combining weak classifiers.

Abstract: Maximum margin Bayesian networks (MMBN) can be trained by solving a convex optimization problem using, for example, interior point (IP) methods (Guo et al., 2005). However, for large datasets this training is computationally expensive (in terms of runtime and memory requirements). Therefore, we propose a less resource intensive batch method to approximately learn a MMBN classifier: we train a set of (weak) MMBN classifiers on subsets of the training data, and then exploit the convexity of the original optimization problem to obtain an approximate solution, i.e., we determine a convex combination of the weak classifiers.
In experiments on different datasets we obtain similar results as for optimal MMBN determined on all training samples. However, in terms of computational efficiency (runtime) we are faster and the memory requirements are much lower. Further, the proposed method facilitates parallel implementation.

1 INTRODUCTION

Endowing machine learning algorithms with the ability to generalize from prior observations is a difficult problem in general. However, in the case of discriminative classifiers, the support vector machine (SVM) is a well established tool with good generalization properties. The ability to generalize is achieved by separating samples from distinct classes by a hyperplane that maximizes the margin between them.

While the large margin concept in SVMs was first suggested by Vladimir Vapnik (Vapnik, 1998) in the 1960s, this idea of margin maximization has only recently been introduced to Bayesian networks by Guo et al. (Guo et al., 2005). In their paper they provided the basic definition of the margin in a probabilistic environment as well as the formulation of a convex optimization problem for learning maximum margin Bayesian networks (MMBN). In contrast to SVMs, for which lots of (efficient) training algorithms have been proposed, e.g., (Platt, 1999; Shalev-Shwartz et al., 2007) and many others, there is only little literature on training MMBNs. Only two approaches are known in the community: the convex optimization approach proposed by Guo et al. (Guo et al., 2005) and a conjugate gradient based approach by Pernkopf et al. (Pernkopf et al., 2011). While the former approach provides slightly better classification rates, it is computationally costly in terms of runtime and memory consumption on large datasets when directly solved by, e.g., interior point methods (Boyd and Lieven, 2004). Despite the lack of literature, MMBNs are serious competitors to SVMs as they allow for the incorporation of domain knowledge as well as efficient handling of missing features and show comparable classification performance in various experiments (Pernkopf et al., 2011).

In this paper we aim to present an algorithm that approximately solves the convex formulation of the MMBN learning problem, leading to good classification results while being computationally less expensive than the original formulation. This is achieved by splitting the training set into subsets and subsequently training an MMBN on each of the subsets. These MMBNs typically exhibit lower classification performance than an MMBN trained on the whole training set. However, the MMBNs can be combined, by exploiting properties of the underlying optimization problem, to form a stronger classifier. In experiments we compare the performance of this approach with the performance of MMBNs trained on the whole training set either by the convex formulation or by the conjugate gradient based approach. In most cases the classification rates of all three methods are competitive. Further experiments demonstrate that the combined classifiers are naturally less prone

Tschiatschek S. and Pernkopf F. (2012).
CONVEX COMBINATIONS OF MAXIMUM MARGIN BAYESIAN NETWORK CLASSIFIERS.
In *Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods*, pages 69-77
DOI: 10.5220/0003770300690077

to overfitting and that they achieve good performance on different partitions of the training set.

Many general schemes for combining classifiers like Adaboost (Freund and Schapire, 1995), Bagging (Breiman, 1996), Bayesian Model Averaging (Bishop, 2007) exist. However, the basic principle is quite different to our approach. While, for example, Adaboost is based on the consecutive training of classifiers, each newly build classifier favoring the missclassified samples of previous classifiers, our approach is simply based on the properties of convex optimization problems.

This paper is organized as follows: In Sect. 2 we introduce the notation and briefly review the basic concepts of Bayesian network classifiers. Based on this, we introduce maximum margin Bayesian network learning in Sect. 3. Subsequently, in Sect. 4 we propose a batch algorithm to reduce the computational and memory complexity of the training process. Section 5 provides empirical results of the proposed algorithm. Finally, Sect. 6 concludes the paper.

2 BAYESIAN NETWORK CLASSIFIERS

A *Bayesian network structure* (BNS) (Koller and Friedman, 2009) is a directed acyclic graph (DAG) $\mathcal{G} = (V, E)$ consisting of nodes V and edges E. The nodes represent random variables (RVs) $Z_0, \ldots, Z_K$ and the edges encode the dependencies between them. A joint probability distribution p over the same RVs *factorizes* according to $\mathcal{G}$, annotated as $p \sim \mathcal{G}$, if it satisfies

$$p(Z_0, \ldots, Z_K) = \prod_{i=0}^{K} p(Z_i | \Pi^{\mathcal{G}}(Z_i)), \qquad (1)$$

where $\Pi^{\mathcal{G}}(Z_i)$ denotes the set of parents of Z_i in $\mathcal{G}$ (Koller and Friedman, 2009). All RVs are assumed to take only discrete values from a finite set.

The tuple $\mathcal{B} = (\mathcal{G}, p)$ is called a *Bayesian network* (BN) if $p \sim \mathcal{G}$ and p is specified by conditional probability distributions (CPDs) associated with the nodes of $\mathcal{G}$ (Koller and Friedman, 2009). If it is not clear from the context, we will annotate the graph $\mathcal{G}$ as $\mathcal{G}^{\mathcal{B}}$ and the probability distribution p as $p^{\mathcal{B}}$, to uniquely identify these two objects with the corresponding BN $\mathcal{B}$.

A Bayesian network employed for classification is a *Bayesian network classifier*. In such networks, without loss of generality, let Z_0 represent the *class* variable $C \in \mathcal{C} = \{1, \ldots, |\mathcal{C}|\}$, where $|\mathcal{C}|$ is the number of classes. The other variables $Z_1, \ldots, Z_K$ are the *attributes* of the classifier. Each of these attributes Z_i can take values in $\{1, \ldots, |Z_i|\}$. The random vector $\mathbf{Z}$ consists of the attributes of the classifier, i.e., $\mathbf{Z} = [Z_1, \ldots, Z_K]$. An instantiation of $\mathbf{Z}$ is denoted by $\mathbf{z}$ and is a specific assignment of values to the random variables. Bold face letters refer to sets of variables while regular letters denote single variables. The CPDs associated with the nodes of $\mathcal{G}$ describing the probability distribution p can be parameterized by a vector θ with entries $\theta^j_{i|h}$ denoting specific conditional probabilities. That is,

$$\theta^j_{i|h} = p(Z_j = i | \Pi^{\mathcal{G}}(Z_j) = h),$$

where $\Pi^{\mathcal{G}}(Z_j) = h$ means that the parents of random variable Z_j take their h^{th} assignment (lexicographically ordered). To make the connection between p and θ explicit, we will typically append the subscript θ to p, i.e., p_θ. Given an unlabeled sample $\mathbf{z}$ the class is predicted as the maximum aposteriori (MAP) estimate, i.e., as

$$\arg\max_{c \in \mathcal{C}} p_\theta(C = c | \mathbf{Z} = \mathbf{z}) = \arg\max_{c \in \mathcal{C}} p_\theta(C = c, \mathbf{Z} = \mathbf{z}), \qquad (2)$$

where the last equality follows because the normalization term $\sum_{\mathbf{z}} p_\theta(\mathbf{Z} = \mathbf{z})$ is constant for all classes.

To employ BNs for classification two problems must be solved:

1. *Structure Learning:* Identify BNSs that appropriately model the dependencies in the considered data and facilitate discrimination. This resorts to discrete optimization problems.
2. *Parameter Learning:* Given a BNS, determine probability distributions that factorize according to the structure and yield good classification results. Parameter learning gives rise to continuous optimization problems.

These problems can be solved independently or jointly. In this paper we do not solve the first problem, but assume a fixed BNS to be given. Anyway, the interested reader can find additional information on this, e.g., in (Acid et al., 2005).

Parameter Learning

There are two different approaches to parameter learning in BNs, namely *generative* and *discriminative* ones. To briefly describe both approaches, let $\mathcal{G}$ be a fixed BNS and $\mathcal{T} = \{(c^{(n)}, \mathbf{z}^{(n)}) : 1 \leq n \leq N\}$ a training set consisting of N i.i.d. labeled training samples, i.e., N instances of values of the RVs represented by the nodes of $\mathcal{G}$.

Generative Parameter Learning. The goal of generative parameter learning is to identify probability distributions $p \sim \mathcal{G}$ that model the joint probability of the classifier attributes and the corresponding class label appropriately. A standard method to find such a distribution, is *maximum likelihood* parameter learning (Pearl, 1988), where p is determined as

$$p^{\mathcal{B}} = \arg\max_{p' \sim \mathcal{G}} \prod_{n=1}^{N} p'(c^{(n)}, \mathbf{z}^{(n)}). \tag{3}$$

Discriminative Parameter Learning. Discriminative approaches aim at learning the class posterior probability $p(c|\mathbf{z})$ directly. One representative of discriminative parameter learning approaches is *conditional likelihood* (CL) parameter learning that is tightly connected to minimizing empirical risk (Greiner et al., 2005). In CL learning the probability distribution p is determined as

$$p^{\mathcal{B}} = \arg\max_{p' \sim \mathcal{G}} \prod_{n=1}^{N} p'(c^{(n)}|\mathbf{z}^{(n)}), \tag{4}$$

i.e., the conditional likelihood of the classes given the attributes is maximized over the samples in the training set.

Another objective for discriminative parameter learning is the maximum margin criterion. It is described in detail in the next section. Generally, discriminative scores such as the margin or CL are not decomposable as the likelihood for maximum likelihood learning. Consequently, there is no closed form solution for discriminative parameter learning and iterative optimization tools are required.

3 MAXIMUM MARGIN BAYESIAN NETWORKS

The idea behind maximum margin parameter learning is to identify a probability distribution p such that the minimal *separation* between samples from different classes is maximized. This approach is inspired by SVMs, for which one tries to maximize the margin between samples from different classes. In the case of SVMs the margin is typically the distance between the feature space representation of the considered samples in some appropriate norm. For Bayesian networks, following the approach taken in (Guo et al., 2005), the margin can be defined as a ratio of probabilities (also named *conditional likelihood ratio*):

Definition 1. *The* multi-class margin *of the labeled sample* $(c, \mathbf{z})$ *in the Bayesian Network* $\mathcal{B}$ *is*

$$d^{\mathcal{B}}(c, \mathbf{z}) = \min_{c' \in C, c' \neq c} \frac{p^{\mathcal{B}}(c|\mathbf{z})}{p^{\mathcal{B}}(c'|\mathbf{z})} = \min_{c' \in C, c' \neq c} \frac{p^{\mathcal{B}}(c, \mathbf{z})}{p^{\mathcal{B}}(c', \mathbf{z})}. \tag{5}$$

Informally, the margin measures how much more likely the sample belongs to the correct class c than to the most likely competing class.

Using this definition, a MMBN can be defined as a BN that maximizes the minimum margin between any two samples from different classes of the training set.

Definition 2. *Let* $\mathcal{T}$ *be a given training set with* N *training samples and* $\mathcal{G}$ *a given BNS. Then,* $\mathcal{B} = (\mathcal{G}, p)$ *is a* maximum margin Bayesian network *if* $\mathcal{B}$ *is a BN and* p *is an optimal solution of the problem*

$$\underset{p' \sim \mathcal{G}}{\text{maximize}} \quad \min_{n=1}^{N} d^{\mathcal{B}}(c^{(n)}, \mathbf{z}^{(n)}). \tag{6}$$

There exists only little literature dealing with the problem of learning MMBNs. Pernkopf et al. (Pernkopf et al., 2011) solved it using a conjugate gradient based method while Guo et al. (Guo et al., 2005) reformulated the margin optimization as a convex optimization problem. The former method is superior in terms of computation speed and memory requirements, but the convex formulation results in slightly better classifiers (Pernkopf et al., 2011). Therefore, it is desireable to solve the convex optimization problem, at least approximately, in a resource-saving way. We present an approach to this in Sect. 4.

Convex Formulation of Maximum Margin Bayesian Networks

The maximum margin learning problem is formulated as a convex optimization problem by introducing a parameter vector $\mathbf{w}$ with elements $w^j_{i|h} = \log(\theta^j_{i|h})$ (in some arbitrary, but fixed order). Using the same order of the elements, the feature vectors $\phi(c^{(n)}, \mathbf{z}^{(n)})$ with elements

$$u^{j,n}_{i|h} = \mathbf{1}_{\{z^{(n)}_j = i \text{ and } \Pi^{\mathcal{G}}(Zj)^{(n)} = h\}},$$

can be defined. The symbol $\mathbf{1}_{\{cond\}}$ denotes the indicator function, i.e., it equals 1 if and only if *cond* is true and 0 otherwise. Then, the probability of any sample $(c^{(n)}, \mathbf{z}^{(n)})$ of the BN can be written as

$$p_\theta(c^{(n)}, \mathbf{z}^{(n)}) = \exp(\phi(c^{(n)}, \mathbf{z}^{(n)})^T \mathbf{w}),$$

where $(\cdot)^T$ denotes the transposition operator. Using this, the logarithm of the multi-class margin of the n^{th}

sample from the training set in (5) becomes

$$\log d^{\mathcal{B}}(c^{(n)},\mathbf{z}^{(n)}) = \min_{c\in C, c\neq c^{(n)}} \left[\phi(c^{(n)},\mathbf{z}^{(n)}) - \phi(c,\mathbf{z}^{(n)})\right]^T \mathbf{w}. \quad (7)$$

Hence, the criterion given in (6) can be reformulated as

$$\begin{aligned} \underset{\gamma,\mathbf{w}}{\text{maximize}} \quad & \gamma && (8)\\ \text{s.t.} \quad & \Delta_{n,c}\mathbf{w} \geq \gamma, \quad \forall n \text{ and } c \neq c^{(n)} && (9)\\ & \sum_{i=1}^{|Z_j|} \exp(w^j_{i|h}) = 1, \quad \forall j,h && (10)\\ & \gamma \geq 0, && (11) \end{aligned}$$

where γ is the logarithm of the minimum of all sample margins and

$$\Delta_{n,c} = \left[\phi(c^{(n)},\mathbf{z}^{(n)}) - \phi(c,\mathbf{z}^{(n)})\right]^T.$$

The constraints (9) ensure that all sample margins are larger than γ. Together with the constraint $\gamma \geq 0$ it is explicitly required that all sample margins are larger than one (zero in logarithmic scale), i.e., all samples are classified correctly. This can only be achieved for separable data. For non-separable data the above problem is infeasible. The constraints (10) ensure that $\mathbf{w}$ describes a valid probability distribution, i.e., all conditional probabilities in the BN sum to 1. Note that the dependency of $\mathbf{w}$ on θ was dropped. In this way, solving the above problem leads to a vector $\mathbf{w}$ that describes a probability distribution p such that $p \sim \mathcal{G}$ and $\mathcal{B} = (\mathcal{G}, p)$ is an MMBN network.

The above problem is still not convex because of the constraints (10). However, these constraints can be relaxed by replacing the equality by an inequality resulting in a convex problem. Further, one slack variable ε_n for each training sample $(c^{(n)},\mathbf{z}^{(n)})$ to account for non-separable data is introduced. Rewriting the objective, we obtain the optimization problem (more details are given in the paper (Guo et al., 2005))

$$\begin{aligned} \underset{\gamma,\varepsilon_1,\ldots,\varepsilon_N,\mathbf{w}}{\text{minimize}} \quad & \frac{1}{2\gamma^2} + B\sum_{n=1}^{N}\varepsilon_n && (12)\\ \text{s.t.} \quad & \Delta_{n,c}\mathbf{w} \geq \gamma - \varepsilon_n, \quad \forall n \text{ and } c \neq c^{(n)} && (13)\\ & \sum_{i=1}^{|Z_j|} \exp(w^j_{i|h}) \leq 1, \quad \forall j,h && (14)\\ & \varepsilon_n \geq 0, \quad \forall n, && (15)\\ & \gamma \geq 0, && (16) \end{aligned}$$

where $B \geq 0$ is a parameter to control the slack-effect, i.e., a tradeoff parameter between a large margin and large sample slacks (similar to the parameter C in SVMs). We also refer to B as the regularization parameter.

Because of the relaxed constraints

$$\sum_{i=1}^{|Z_j|} \exp(w^j_{i|h}) \leq 1, \quad \forall j,h,$$

in (12) the vector $\mathbf{w}$ of an optimal solution does not necessarily describe valid CPDs. That is, $\mathbf{w}$ can represent a *subnormalized* probability distribution (abusing terminology). However, for some network structures, e.g., Naive Bayes and Tree Augmented Networks, the resulting parameter vector $\mathbf{w}$ allows for renormalization without changing the decision function in (2). For more details we refer the interested reader to (Roos et al., 2005).

4 CONVEX COMBINATION OF WEAK MMBNS

4.1 Discussion of Sample/Feature Size

Equation (12) represents a convex optimization problem that can be solved by any minimization method allowing for a nonlinear objective and nonlinear convex inequality constraints. For example, interior point methods (Boyd and Lieven, 2004) are such a class of methods. While these methods are efficient in theory, Pernkopf et al. (Pernkopf et al., 2011) observed large computational requirements when learning MMBNs parameters using the convex formulation. This is especially true for large training sets in terms of samples and/or features:

- The total number of unknowns in the optimization problem is $1+N+\text{Var}(\mathcal{G})$, where $\text{Var}(\mathcal{G})$ denotes the number of variables required to fully specify the conditional probability tables associated with the nodes of $\mathcal{G}$ in a BN. Hence, $\text{Var}(\mathcal{G})$ is large for dense network structures and structures with many RVs and/or large cardinalities of the RVs.

  Large values of $1+N+\text{Var}(\mathcal{G})$ lead to high dimensional search spaces, typically resulting in long runtimes for the optimization process.

- Despite the non-negativity constraints on γ and $\varepsilon_1,\ldots,\varepsilon_N$, there is one nonlinear inequality constraint for each conditional probability of the network. Additional, for every training sample $|C|$ linear inequality constraints are required.

  The number of inequalities of the convex formulation influences the required number of iterations

for interior point methods to converge to a solution with specific accuracy (Boyd and Lieven, 2004). Larger numbers of inequality constraints lead to larger number of iterations.

4.2 Proposed Batch Strategy

To reduce the computational burden we propose to apply a batch algorithm. The idea behind the algorithm is to train a sequence of weak classifiers, each on a subset of the training set using the convex formulation in (12). Subsequently, these classifiers are combined to form a strong classifier.

Given a BNS $\mathcal{G}$, the number $M \in \mathbb{N}$ of weak classifiers to train and a regularization parameter $B \geq 0$, the algorithm performs the following steps:

1. Determine a cover $\overline{\mathcal{T}} = \{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$ of $\mathcal{T}$, i.e., $\overline{\mathcal{T}}$ is comprised of M subsets of the training set such that
$$\mathcal{T} = \bigcup_{m=1}^{M} \mathcal{T}_m.$$
2. Train M classifiers on the partial training sets, i.e., determine $\gamma_m, \varepsilon_m, \mathbf{w}_m$ as solutions to (12) using training set $\mathcal{T}_m$.
3. Determine an optimal convex combination of the M classifiers such that the original objective is minimized. Therefore, determine $\alpha_1, \ldots, \alpha_M$ as the optimization variables that solve
$$\underset{\alpha'_1,\ldots,\alpha'_M}{\text{minimize}} \quad \text{MMBN}\left(\sum_{m=1}^{M} \alpha'_m \mathbf{w}_m\right) + R, \tag{17}$$
$$\text{s.t.} \quad \alpha'_1, \ldots, \alpha'_M \geq 0, \tag{18}$$
$$\alpha'_1 + \ldots + \alpha'_M = 1, \tag{19}$$
where
$$R := D\sqrt{\sum_{m=1}^{M}\left(\alpha_m - \frac{1}{M}\right)^2} \tag{20}$$
and $\text{MMBN}(\mathbf{w})$ is the minimization problem (12) with fixed $\mathbf{w}$, i.e.,
$$\underset{\gamma,\varepsilon_1,\ldots,\varepsilon_N}{\text{minimize}} \quad \frac{1}{2\gamma^2} + B\sum_{n=1}^{N}\varepsilon_n \tag{21}$$
$$\text{s.t.} \quad \Delta_{n,c}\mathbf{w} \geq \gamma - \varepsilon_n, \quad \forall n \text{ and } c \neq c^{(n)}$$
$$\gamma \geq 0, \quad \sum_{i=1}^{|Z_j|} \exp(w^j_{i|h}) \leq 1, \quad \forall j, h,$$
$$\varepsilon_n \geq 0, \quad \forall n.$$
The parameter $D \geq 0$ can be used to ensure that all weak classifiers participate in the final classifier. Selecting large values for D results in all weak classifiers being equally weighted when constructing the final classifier. Hence, the second term in (17) can be viewed as a regularization term.

The constraints on the coefficients $\alpha'_1, \ldots, \alpha'_M$, i.e., nonnegativity (18) and sum to one (19), are required for *convex* combinations of $\mathbf{w}_1, \ldots, \mathbf{w}_M$. In this way the *convex hull* defined as the set of all convex combinations of $\mathbf{w}_1, \ldots, \mathbf{w}_M$ is searched for an optimal parameter vector $\mathbf{w}'$ in the sense of (21). Considering not arbitrary linear combinations of $\mathbf{w}_i$ but only convex combinations ensures that the vector $\mathbf{w}' = \sum_{m=1}^{M} \alpha'_i \mathbf{w}_i$ satisfies the subnormalization constraints of the original convex formulation.

Equation (21) essentially describes a one-dimensional optimization problem and can be solved easily. Details can be found in the Appendix.

4. Return the BN specified by $\mathcal{B} = (\mathcal{G}, p(\mathbf{w}))$, where
$$\mathbf{w} = \sum_{m=1}^{M} \alpha_m \mathbf{w}_m. \tag{22}$$
and $p(\mathbf{w})$ is the probability distribution obtained from $\mathbf{w}$ by renormalization.

4.3 Geometric Interpretation of the Batch Algorithm

Figure 1 illustrates the principle of the proposed batch algorithm in a two dimensional parameter space for $D = 0$. The parameters of an optimal classifier in the sense of (12) are indicated by a plus sign and the parameters of the weak classifiers found by splitting the overall training set into four subsets are shown as grey circles. The objective to be minimized is visualized by contour lines, where the optimal classifier is able to detect the global optimum. Solving (17) corresponds to searching the gray shaded region, i.e., searching the convex hull spanned by the parameters of the weak classifiers. In the shown example, the parameters marked by the diamond are the optimal ones in the convex hull, i.e., the parameters with the minimal objective.

If the parameters of the optimal classifier would lie in the convex hull, then the proposed algorithm would find the optimal solution. While it seems reasonable that this occurs in low dimensional parameter spaces, this will typically not happen in high dimensional spaces; in the case of the USPS database we have a ~ 3000 dimensional parameter space. Having trained, for example, 40 weak classifiers only a small

subset of the parameter space lies in the spanned convex hull making it very *unlikely* that a global minimum is found. Clearly, the performance of the proposed batch algorithm depends on the selected cover and on the determined weak classifiers. Training of the weak classifiers such that the spanned region covers good classifiers is subject to future work.

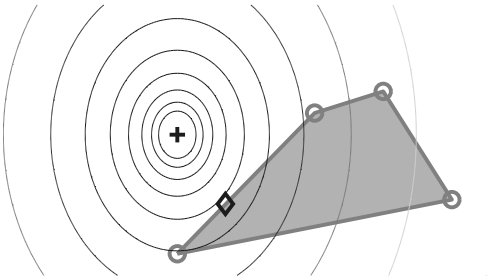


Figure 1: Geometric interpretation of the proposed batch algorithm.

5 EXPERIMENTS

We present classification results for frame-based phonetic data using the TIMIT speech corpus and for handwritten digit recognition using the USPS and MNIST datasets. The experiments were performed using a fixed Bayesian network structure, namely a naive Bayes (NB) classifier structure. The NB structure is illustrated in Fig. 2. All attributes $Z_1, \ldots, Z_K$ are conditionally independent given the class. While the structure is simple, good performance can be achieved in various applications even if the conditional independence assumptions are unrealistic or even false in most of the data (Rish, 2001).

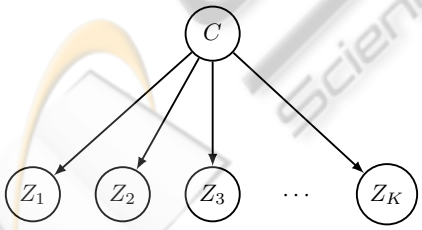


Figure 2: Naive Bayes network.

5.1 Simulation Setup

We used IPOPT (Interior Point OPtimizer) (Wächter and Biegler, 2005), a freely available software package for large-scale optimization that showed good performance in various applications to solve the optimization problem (12). IPOPT requires a linear solver to compute the step-directions in the applied interior point method. In our experiments, we employed PARDISO (Schenk and Gärtner, 2004; Schenk and Gärtner, 2006; Karypis and Kumar, 2006) for this purpose. The optimization problem (17) was solved using the function `fmincon` included in the optimization toolbox from Matlab. Furthermore, the cover of the training set in the batch algorithm was determined as follows: The training set was split into M equally sized disjoint subsets. The samples in each subset were selected according to the prior class probabilities. CPU time experiments were performed on a personal computer with 2.8 GHz, 32 GB of memory and exploiting (multicore) parallelization of up to 11 cores. The parallelization especially results in a speedup of the linear solver PARDISO. The reported CPU times are those reported by IPOPT and Matlab. The regularization parameter B was selected using cross tuning for the TIMIT data, while fixed as $B = 1$ for the USPS and MNIST data.

5.2 Datasets

In our experiments we considered the MNIST, USPS and TIMIT databases, that we briefly describe in the following:

- **TIMIT-4/6 Data.** This dataset 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 phonetic 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, each with 4 and 6 classes). The data have been split into two mutually exclusive subsets where 70 % of the data is used for training and 30 % for testing. More details about the features can be found in (Pernkopf et al., 2009).
- **USPS Data.** This dataset contains 11000 uniformly distributed handwritten digit images from zip codes of mail envelopes. The database is split into 8000 images for training and 3000 for testing. Each digit is represented as a 16×16 grayscale image, where each pixel is considered as feature.
- **MNIST Data (LeCun et al., 1998).** This dataset contains 60000 samples for training and 10000 samples for testing of handwritten digits. To reduce computation time we reduced the training set to 10000 samples according to the prior class

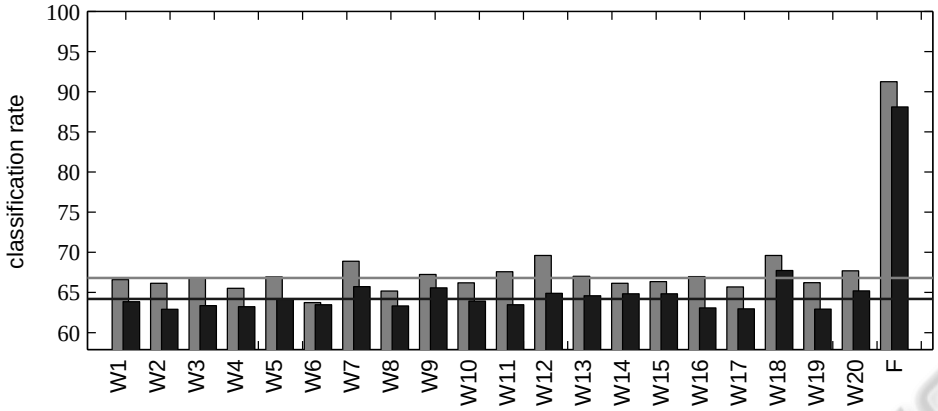


Figure 3: Classification performance in [%] of the proposed batch algorithm on the USPS dataset. The classification rates on the complete training set (*light grey bars*) and on the test set (*dark grey bars*) of the weak classifiers (*labeled as "W1"*, ..., *"W20"*) and of the final classifier (*labeled as F*) are shown. The *light gray horizontal line* marks the average classification rate of the weak classifiers on the training set and the *dark gray horizontal line* the average classification rate on the test set, respectively.

Table 1: Classification rates CR in [%] for different datasets and number of splits M. The columns CT shows the total CPU time for the parameter learning, the column $\mathrm{CT}_{\max}$ the maximum CPU time for learning a single weak classifier and $\mathrm{CT}_{\mathrm{conv}}$ the CPU time for finding the optimal convex combination of the weak classifiers (all times are given in seconds).

Database	M	CT	CR	M	B	D	$\mathrm{CT}_{\max}$	$\mathrm{CT}_{\mathrm{conv}}$	CT	CR
MNIST	1	20152	86.01	100	1	$1\cdot10^5$	52	2180	5380	83.99
USPS	1	89364	90.90	20	1	$1\cdot10^3$	1494	388	25599	88.10
TIMIT Ma+Fe-4	1	3811	92.23	40	$4\cdot10^{-3}$	$1\cdot10^3$	105	1115	4315	92.05
TIMIT Ma-4	1	2576	93.06	20	$4\cdot10^{-3}$	$1\cdot10^3$	51	230	1072	92.95
TIMIT Fe-4	1	1502	91.82	20	$4\cdot10^{-3}$	$1\cdot10^3$	55	310	1296	91.64
TIMIT Ma+Fe-6	1	19574	85.61	40	$4\cdot10^{-3}$	$1\cdot10^6$	612	1432	21952	85.45
TIMIT Ma-6	1	10285	86.67	20	$4\cdot10^{-3}$	$1\cdot10^6$	207	240	3886	86.11
TIMIT Fe-6	1	9676	85.36	20	$4\cdot10^{-3}$	$7\cdot10^5$	214	503	3912	85.13

probabilities. The digits respresented by gray-level images were down-sampled by a factor of two resulting in a resolution of 14×14 pixels, i.e., 196 features.

5.3 Classification Results on the USPS Data

We applied the proposed batch algorithm on the USPS dataset by training 20 weak classifiers. The regularization parameter was selected as $B=1$ (without applying any parameter-tuning method). The performance in terms of the classification rate of the weak classifiers on the complete training and the test set is shown in Fig. 3. Additionally, the performance of the final classifier determined by solving (17) is shown.

The final classifier has a 15 to 25 percent better classification rate than the weak classifiers (on the training, as well as, on the test set).[1] All of the weak classifiers achieved a classification rate of 100 percent on the training set $\mathcal{T}_m$ they were trained on (not shown in the figure) indicating overfitting. In contrast to this, the final classifier avoids this problem. This is also observed in other algorithms that combine the results of several classifiers like bagging and boosting (Breiman, 1996; Freund and Schapire, 1995).

5.4 Classification Results for Various Databases

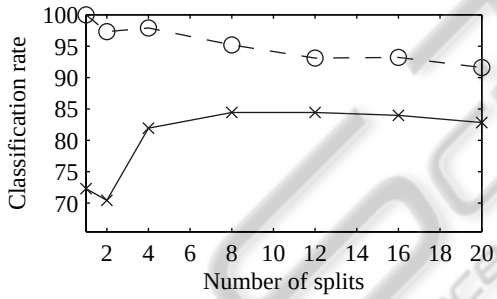
Classification results for the TIMIT, USPS and MNIST database are presented in Tab. 1. Further, the required CPU times for the parameter learning are shown. Note that running the proposed batch algorithm with $M=1$ is equal to solving the original convex optimization problem (12). Hence, classifiers trained by the proposed algorithm perform slightly

[1] However, note that the performance gain of the final classifier is not always as impressive as in this case, e.g., on the order of 3 to 10 percent on the TIMIT datasets.

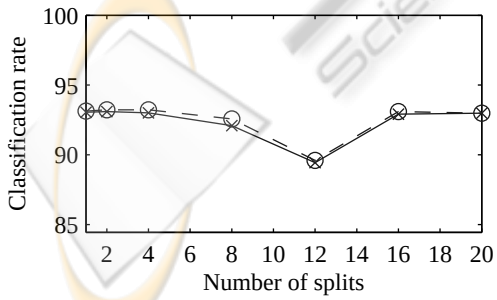
worse than classifiers trained on the whole training sets. Nevertheless, the classification rates are competitive.

5.5 Classification Results for Different Partitionings

For this experiment the USPS and TIMIT Ma-4 data was considered. The number of training samples of the whole training set was reduced to 1000 samples in the case of the USPS data. The parameters B and D were selected as in Tab. 1. Classification results for the two datasets and for varying number of splits M of the training set are shown in Fig. 4. For USPS data the classification rate on the training set decreases with an increasing number of splits while the classification rate on the test set increases. This indicates overfitting for low values of M. For TIMIT data a region ($M = 12$ in the plot) can be identified in which both the classification rate on the training and on the test set decreases. For this region, the convex hull of the determined weak classifiers does not contain a strong classifiers for the complete training set. Clearly, to achieve optimal performance for all different partitionings the parameters B and D would have to be retuned for every partitioning.



(a) USPS data (only 1000 training samples used)



(b) TIMIT-4/6 Ma-4 database

Figure 4: Classification rate in [%] on the training set (*dashed line*) and on the test set (*solid line*) for different numbers of splits of the training set.

5.6 Comparison to Conjugate Gradient MMBN and SVMs

Table 2 compares the classification results of the proposed algorithm, the conjugate gradient MMBN algorithm (CG-MMBN) (Pernkopf et al., 2011) and SVMs (with fixed parameters $C^* = 1, \sigma = 0.05$) on the TIMIT-4/6 data. SVMs slightly outperform the MMBNs in all cases. The proposed algorithm achieves approximately the same classification rates as CG-MMBN.

Table 2: Classification rates CR in percent for different TIMIT-4/6 datasets and different classifiers. The columns M and D show the parameters supplied to the proposed batch algorithm. The parameter $B = 4 \cdot 10^{-3}$ in all experiments.

	proposed MMBN			CG-MMBN	SVM
Database	M	D	CR	CR	CR
Ma+Fe-4	40	$1 \cdot 10^3$	92.05	92.09	92.49
Ma-4	20	$1 \cdot 10^3$	92.95	92.97	93.30
Fe-4	20	$1 \cdot 10^3$	91.64	91.57	92.14
Ma+Fe-6	40	$1 \cdot 10^6$	85.45	85.41	86.24
Ma-6	20	$1 \cdot 10^6$	86.11	86.20	87.19
Fe-6	20	$7 \cdot 10^5$	85.13	84.85	86.19

6 CONCLUSIONS

We proposed a batch method for approximately learning maximum margin Bayesian networks using a convex formulation. The method is less computationally demanding than solving the original formulation. Still, the obtained results are comparable, as demonstrated in various experiments. Further, the method facilitates parallel implementation as training the weak classifiers could be fully parallelized.

Since the presented results are quite promising, we plan to address the following problems:

- Derivation of generalization bounds for MMBNs trained by the convex combination scheme.
- Does an optimal cover C of $\mathcal{T}$ exist, i.e., a cover that will result in a best possible classifier?
- Performing further experiments, especially on more general network structures.

ACKNOWLEDGEMENTS

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

REFERENCES

Acid, S., Campos, L. M., and Castellano, J. G. (2005). Learning Bayesian network classifiers: Searching in a space of partially directed acyclic graphs. *Machine Learning*, 59:213–235.

Bishop, C. M. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing edition.

Boyd, S. and Lieven, V. (2004). *Convex Optimization*. Cambridge University Press.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Freund, Y. and Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting.

Greiner, R., Su, X., Shen, B., and Zhou, W. (2005). Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Machine Learning*, 59(3):297–322.

Guo, Y., Wilkinson, D., and Schuurmans, D. (2005). Maximum margin Bayesian networks. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence*, pages 233–242. AUAI Press.

Karypis, G. and Kumar, V. (2006). A fast and high quality multilevel scheme for partitioning irregular graphs,. *SIAM Journal on Scientific Computing*, 20(1):359–392.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Pernkopf, F., Van Pham, T., and Bilmes, J. A. (2009). Broad phonetic classification using discriminative Bayesian networks. *Speech Communication*, 51(2):151–166.

Pernkopf, F., Wohlmayr, M., and Tschiatschek, S. (2011). Maximum margin bayesian network classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (accepted).

Platt, J. (1999). Sequential minimal optimization: A fast algorithm for training support vector machines. *Advances in Kernel Methods-Support Vector Learning*, pages 1–21.

Rish, I. (2001). An empirical study of the naive Bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, pages 41–46.

Roos, T., Wettig, H., Grünwald, P., Myllymäki, P., and Tirri, H. (2005). On Discriminative Bayesian Network Classifiers and Logistic Regression. *Machine Learning*, 59(3):267–296.

Schenk, O. and Gärtner, K. (2004). Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generation Computer Systems*, 20(3):475–487.

Schenk, O. and Gärtner, K. (2006). On fast factorization pivoting methods for symmetric indefinite systems. *Electronic Transactions on Numerical Analysis*, 23:158–179.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning*, pages 807–814. ACM.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

Wächter, A. and Biegler, L. T. (2005). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

APPENDIX

Solving the Intermediate Optimization Problem

The optimization problem (21) for fixed $\mathbf{w}$ satisfying the subnormalization constraints and given training set $\mathcal{T}$ of N samples can be rewritten as

$$\begin{aligned} \underset{\gamma,\varepsilon_1,\ldots,\varepsilon_N}{\text{minimize}} \quad & \frac{1}{2\gamma^2} + B\sum_{n=1}^{N}\varepsilon_n \\ \text{s.t.} \quad & \widetilde{x}_{n,c} \geq \gamma - \varepsilon_n, \quad \forall n \text{ and } c \neq c^{(n)} \\ & \gamma \geq 0, \quad \varepsilon_n \geq 0, \quad \forall n, \end{aligned} \tag{23}$$

where we set $\widetilde{x}_{n,c} = \Delta_{n,c}\mathbf{w}$. For $n = 1,\ldots,N$ let

$$x_n = \min_{c\in C, c\neq c^{(n)}} \widetilde{x}_{n,c}.$$

Then, the above problem is equivalent to

$$\begin{aligned} \underset{\gamma,\varepsilon_1,\ldots,\varepsilon_N}{\text{minimize}} \quad & \frac{1}{2\gamma^2} + B\sum_{n=1}^{N}\varepsilon_n \\ \text{s.t.} \quad & x_n \geq \gamma - \varepsilon_n, \quad \forall n \\ & \gamma \geq 0, \quad \varepsilon_n \geq 0, \quad \forall n, \end{aligned} \tag{24}$$

because the removed constraints will be simultaneously satisfied. In an optimal solution with margin γ' the term $\sum_{n=1}^{N}\varepsilon_n$ must be as small as possible. Therefore, all the ε_n are required to take the minimal value that is still feasible. This is $\varepsilon_n = \gamma' - x_n$, if this quantity is positive. Or $\varepsilon_n = 0$ otherwise. In this way, the optimization problem becomes

$$\begin{aligned} \underset{\gamma}{\text{minimize}} \quad & \frac{1}{2\gamma^2} + B\sum_{n=1}^{N}\max\{\gamma - x_n, 0\} \\ \text{s.t.} \quad & \gamma \geq 0 \end{aligned} \tag{25}$$

and can be easily solved. If required, the slacks ε_n can subsequently be calculated as $\varepsilon_n = \max\{\gamma - x_n, 0\}$.