Parameter Learning of Bayesian Network Classifiers Under Computational Constraints

Sebastian Tschiatschek[1(✉)] and Franz Pernkopf[2]

[1] Learning and Adaptive Systems Group, ETH Zurich, Zürich, Switzerland
sebastian.tschiatschek@inf.ethz.ch
[2] Signal Processing and Speech Communication Laboratory,
Graz University of Technology, Graz, Austria
pernkopf@tugraz.at

Abstract. We consider online learning of Bayesian network classifiers (BNCs) with reduced-precision parameters, i.e. the conditional-probability tables parameterizing the BNCs are represented by low bit-width fixed-point numbers. In contrast to previous work, we analyze the learning of these parameters using reduced-precision arithmetic only which is important for computationally constrained platforms, e.g. embedded- and ambient-systems, as well as power-aware systems. This requires specialized algorithms since naive implementations of the projection for ensuring the sum-to-one constraint of the parameters in gradient-based learning are not sufficiently accurate. In particular, we present generative and discriminative learning algorithms for BNCs relying only on reduced-precision arithmetic. For several standard benchmark datasets, these algorithms achieve classification-rate performance close to that of BNCs with parameters learned by conventional algorithms using double-precision floating-point arithmetic. Our results facilitate the utilization of BNCs in the foresaid systems.

Keywords: Bayesian network classifiers · Reduced-precision · Resource-constrained computation · Generative/discriminative learning

1 Introduction

Most commonly Bayesian network classifiers (BNCs) are implemented on nowadays desktop computers, where double-precision floating-point numbers are used for parameter representation and arithmetic operations. In these BNCs, inference and classification is typically performed using the same precision for parameters and operations, and the executed computations are considered as exact. However, there is a need for BNCs working with limited computational resources. Such resource-constrained BNCs are important in domains such as ambient computing, on-satellite computations[1] or acoustic environment classification in hearing

F. Pernkopf—This work was supported by the Austrian Science Fund (FWF) under the project number P25244-N15.

[1] Computational capabilities on satellites are still severely limited due to power constraints and restricted availability of hardware satisfying the demanding requirements with respect to radiation tolerance.

© Springer International Publishing Switzerland 2015
A. Appice et al. (Eds.): ECML PKDD 2015, Part I, LNAI 9284, pp. 86–101, 2015.
DOI: 10.1007/978-3-319-23528-8_6

aids, machine learning for prosthetic control, e.g. a brain implant to control hand movements, amongst others. In all these applications, a trade-off between accuracy and required computational resources is essential.

In this paper, we investigate BNCs with limited computational demands by considering BNCs with reduced-precision parameters, i.e. fixed-point parameters with limited precision.[2] Using reduced-precision parameters is advantageous in many ways, e.g. power consumption compared to full-precision implementations can be reduced [20] and reduced-precision parameters enable one to implement many BNCs in parallel on field programmable gate arrays (FPGAs), i.e. the circuit area requirements on the FPGA correlate with the parameter precision [9]. Our investigations are similar to those performed in digital signal-processing, where reduced-precision implementations for digital signal processors are of great importance [10]. Note that there is also increased interest in implementing other machine learning models, e.g. neural networks, using reduced-precision parameters/computations to achieve faster training and to facilitate the implementation of larger models [2,18].

We are especially interested in learning the reduced-precision parameters using as little computational resources as possible. To decide on how to perform this learning, several questions should be answered. Should reduced-precision parameters be learned in a pre-computation step in which we can exploit the full computational power of nowadays computers? Or is it necessary to learn/adopt parameters using reduced-precision arithmetic only? The answers to these questions depend on the application of interest and identify several learning scenarios that are summarized in Figure 1. In the following, we discuss these scenarios briefly:

(a) **Training and testing using full-precision arithmetic.** This corresponds to what machine learners typically do, i.e. all computations are performed using full-precision arithmetic.
(b) **Training using reduced-precision and testing using full-precision arithmetic.** A rash thought rejects this option. But it might be interesting in the vicinity of big-data where the amount of data is so huge that it can only be processed in a compressed form, i.e. in reduced-precision.
(c) **Training using full-precision and testing using reduced-precision arithmetic.** This describes an application scenario where BNCs with precomputed parameters can be used, e.g. hearing-aids for auditory scene classification. This scenario enables one to exploit large computational resources for parameter learning, while limiting computational demands at test time. Recent work considered this for BNCs [22].
(d) **Training and testing using reduced-precision arithmetic.** This is the scenario considered within this paper. It opens the door to many interesting applications, e.g. continuous parameter adaptation in hearing-aids using reduced-precision computations only. Another example could be a

[2] We are interested in fixed-point arithmetic and not in floating-point arithmetic, because typically the implementation of fixed-point processing units requires less resources than the implementation of floating-point processing units.

Testing \ Training	**full-precision**	**reduced-precision**
full-precision	classical scenario, e.g. training and testing on PCs	potentially relevant for big-data applications
reduced-precision	full-precision pre-computation of parameters	e.g. parameter adaptation during testing

Fig. 1. Combinations of training/testing using full-precision/reduced-precision arithmetic.

satellite-based system for remote sensing that tunes its parameter according to changing atmospheric conditions.

We start our investigation of parameter learning using reduced-precision computations by analyzing the effect of approximate computations on *online* parameter learning. This leads to the observation that the approximate projections needed in the used projected gradient ascent/descent algorithms to ensure the sum-to-one normalization constraints of the parameters can severely affect the learning process. We circumvent the need for these projections by proposing special purpose learning algorithms for generative maximum likelihood (ML) and discriminative maximum margin (MM) parameters.

This paper is structured as follows: In Section 2 we consider related work, followed by an introduction of the used notation and some background on parameter learning in Bayesian networks (BNs) in Section 3. We derive our proposed algorithms in Section 4 and test them in experiments in Section 5. In Section 6 we conclude the paper.

2 Related Work

For undirected graphical models, approximate inference *and* learning using integer parameters has been proposed [16]. While undirected graphical models are more amenable to integer approximations mainly due to the absence of sum-to-one constraints, there are domains where probability distributions represented by directed graphical models are desirable, e.g. in expert systems in the medical domain.

Directly related work can be summarized as follows:

- The *feasibility of BNCs with reduced-precision floating-point parameters* has been empirically investigated in [14,24]. These papers analyzed (i) the effect of precision-reduction of the parameters on the classification performance of BNCs, and (ii) how BNCs with reduced-precision parameters can be implemented using integer computations only.
- The above mentioned experimental studies where extended by a thorough theoretical analysis of using fixed-point parameters in BNCs [23]. The authors used fixed-point numbers for the following two reasons: First, because fixed-point parameters can even be used on computing platforms without floating-point processing capabilities. Second, because summation of fixed-point numbers is exact (neglecting the possibility of overflows), while summation of floating-point numbers is in general not exact.
  In particular, theoretical bounds on the classification performance when using reduced-precision fixed-point parameters have been analyzed in [21, 23]. The authors derived worst-case and probabilistic bounds on the classification rate (CR) for different bit-widths. Furthermore, they compared the classification performance and the robustness of BNCs with generatively and discriminatively optimized parameters, i.e. parameters optimized for high data likelihood and parameters optimized for classification, with respect to parameter quantization.
- In [22], *learning of reduced-precision parameters using full-precision computations* was addressed while the work mentioned above considers only rounding of double-precision parameters. An algorithm for the computation of MM reduced-precision parameters was presented and its efficiency was demonstrated. The resulting parameters had superior classification performance compared to parameters obtained by simple rounding of double-precision parameters, particularly for very low numbers of bits.

3 Background and Notation

Probabilistic Classification. Probabilistic classifiers are embedded in the framework of probability theory. One assumes a random variable (RV) C denoting the class and RVs $X_1, \ldots, X_L$ representing the attributes/features of the classifier. Each X_i can take one value in the set $\mathbf{val}(X_i)$. Similarly, C can assume values in $\mathbf{val}(C)$, i.e. $\mathbf{val}(C)$ is the set of classes. We denote the random vector consisting of $X_1, \ldots, X_L$ as $\mathbf{X} = (X_1, \ldots, X_L)$. Instantiations of RVs are denoted using lower case letters, i.e. $\mathbf{x}$ is an instantiation of $\mathbf{X}$ and c an instantiation of C, respectively. The RVs $C, X_1, \ldots, X_L$ are assumed to be jointly distributed according to the distribution $\mathrm{P}^*(C, \mathbf{X})$. In typical settings, $\mathrm{P}^*(C, \mathbf{X})$ is unknown, but a number of samples drawn iid from $\mathrm{P}^*(C, \mathbf{X})$ is at hand, i.e. a training set $\mathcal{D} = ((c^{(n)}, \mathbf{x}^{(n)}) \,|\, 1 \leq n \leq N)$, where $c^{(n)}$ denotes the instantiation of the RV C and $\mathbf{x}^{(n)}$ the instantiation of $\mathbf{X}$ in the n^{th} training sample. The aim is to induce *good* classifiers provided the training set, i.e. classifiers with low generalization error. Any probability distribution $\mathrm{P}(C, \mathbf{X})$

naturally induces a classifier $h_{\mathrm{P}(C,\mathbf{X})}$ according to $h_{\mathrm{P}(C,\mathbf{X})}\colon \mathbf{val}(\mathbf{X}) \to \mathbf{val}(C)$, $\mathbf{x} \mapsto \arg\max_{c' \in \mathbf{val}(C)} \mathrm{P}(c'|\mathbf{x})$. In this way, each instantiation $\mathbf{x}$ of $\mathbf{X}$ is classified by the maximum a-posteriori (MAP) estimate of C given $\mathbf{x}$ under $\mathrm{P}(C, \mathbf{X})$. Note that $\arg\max_{c' \in \mathbf{val}(C)} \mathrm{P}(c'|\mathbf{x}) = \arg\max_{c' \in \mathbf{val}(C)} \mathrm{P}(c', \mathbf{x})$.

Bayesian Networks and Bayesian Network Classifiers. We consider probability distributions represented by BNs [7,11]. A BN $\mathcal{B} = (\mathcal{G}, \mathcal{P}_\mathcal{G})$ consists of a directed acyclic graph (DAG) $\mathcal{G} = (\mathbf{Z}, \mathbf{E})$ and a collection of conditional probability distributions $\mathcal{P}_\mathcal{G} = (\mathrm{P}(X_0|\mathbf{Pa}(X_0)), \ldots, \mathrm{P}(X_L|\mathbf{Pa}(X_L)))$, where the terms $\mathbf{Pa}(X_i)$ denote the set of parents of X_i in $\mathcal{G}$. The nodes $\mathbf{Z} = (X_0, \ldots, X_L)$ correspond to RVs and the edges $\mathbf{E}$ encode conditional independencies among these RVs. Throughout this paper, we often denote X_0 as C, i.e. X_0 represents the class. Then, a BN defines the joint distribution

$$\mathrm{P}^\mathcal{B}(C, X_1, \ldots, X_L) = \mathrm{P}(C|\mathbf{Pa}(C)) \prod_{i=1}^{L} \mathrm{P}(X_i|\mathbf{Pa}(X_i)). \tag{1}$$

According to the joint distribution, a BN $\mathcal{B}$ induces the classifier $h_\mathcal{B} = h_{\mathrm{P}^\mathcal{B}(C,\mathbf{X})}$.

In this paper, we assume discrete valued RVs only. Then, a general representation of $\mathcal{P}_\mathcal{G}$ is a collection of conditional probability tables (CPTs), i.e. $\mathcal{P}_\mathcal{G} = (\boldsymbol{\theta}^0, \ldots, \boldsymbol{\theta}^L)$, with $\boldsymbol{\theta}^i = (\theta^i_{j|\mathbf{h}} | j \in \mathbf{val}(X_i), \mathbf{h} \in \mathbf{val}(\mathbf{Pa}(X_i)))$, where $\theta^i_{j|\mathbf{h}} = \mathrm{P}(X_i = j|\mathbf{Pa}(X_i) = \mathbf{h})$. The BN distribution can then be written as

$$\mathrm{P}^\mathcal{B}(C = c, \mathbf{X} = \mathbf{x}) = \prod_{i=0}^{L} \prod_{j \in \mathbf{val}(X_i)} \prod_{\mathbf{h} \in \mathbf{val}(\mathbf{Pa}(X_i))} {\theta^i_{j|\mathbf{h}}}^{\nu^i_{j|\mathbf{h}}}, \tag{2}$$

where $\nu^i_{j|\mathbf{h}} = \mathbf{1}_{([c,\mathbf{x}](X_i)=j \text{ and } [c,\mathbf{x}](\mathbf{Pa}(X_i))=\mathbf{h})}$.[3] We typically represent the BN parameters in the logarithmic domain, i.e. $w^i_{j|\mathbf{h}} = \log \theta^i_{j|\mathbf{h}}$, $\mathbf{w}^i = (w^i_{j|\mathbf{h}} | j \in \mathbf{val}(X_i), \mathbf{h} \in \mathbf{val}(\mathbf{Pa}(X_i)))$, and $\mathbf{w} = (\mathbf{w}^0, \ldots, \mathbf{w}^L)$. In general, we will interpret $\mathbf{w}$ as a vector, whose elements are addressed as $w^i_{j|\mathbf{h}}$. We define a vector-valued function $\boldsymbol{\phi}(c, \mathbf{x})$ of the same length as $\mathbf{w}$, collecting $\nu^i_{j|\mathbf{h}}$, analog to the entries $w^i_{j|\mathbf{h}}$ in $\mathbf{w}$. In that way, we can express the logarithm of (2) as

$$\log \mathrm{P}^\mathcal{B}(C = c, \mathbf{X} = \mathbf{x}) = \boldsymbol{\phi}(c, \mathbf{x})^T \mathbf{w}. \tag{3}$$

Consequently, classification, can be performed by simply adding the log-probabilities corresponding to an instantiation $[c, \mathbf{x}]$ for all $c \in \mathbf{val}(C)$.[4]

Fixed-Point Numbers. Fixed-point numbers are essentially integers scaled by a constant factor, i.e. the fractional part has a fixed number of digits. We characterize fixed-point numbers by the number of integer bits b_i and the number

[3] Note that $[c, \mathbf{x}]$ denotes the joint instantiation of C and $\mathbf{X}$ and $[c, \mathbf{x}](\mathbf{A})$ corresponds to the subset of values of $[c, \mathbf{x}]$ indexed by $\mathbf{A} \subseteq \{X_0, \ldots, X_L\}$.

[4] In general graphs, potentially with latent variables, the needed inference can be performed using *max-sum* message passing [7,13].

of fractional bits b_f. The addition of two fixed-point numbers can be easily and accurately performed, while the multiplication of two fixed-point numbers often leads to overflows and requires truncation to achieve results in the same format.

Learning Bayesian Network Classifiers

BNs for classification can be optimized in two ways: firstly, one can select the graph structure $\mathcal{G}$ (*structure learning*), and secondly, one can learn the conditional probability distributions $\mathcal{P}_\mathcal{G}$ (*parameter learning*). In this paper, we consider fixed structures of the BNCs, namely naive Bayes (NB) and tree augmented network (TAN) structures [4], i.e. 1-tree among the attributes. The NB structure implies conditional independence of the features, given the class. Obviously, this conditional independence assumption is often violated in practice. TAN structures relax these strong independence assumptions, enabling better classification performance.[5]

Parameter Learning. The conditional probability densities (CPDs) $\mathcal{P}_\mathcal{G}$ of BNs can be optimized either generatively or discriminatively. Two standard approaches for optimizing $\mathcal{P}_\mathcal{G}$ are:

- **Generative Maximum Likelihood Parameters.** In generative parameter learning one aims at identifying parameters modeling the generative process that results in the data of the training set, i.e. generative parameters are based on the idea of *approximating* $\mathrm{P}^*(C, \mathbf{X})$ by a distribution $\mathrm{P}^\mathcal{B}(C, \mathbf{X})$. An example of this paradigm is *maximum likelihood (ML)* learning. Its objective is maximization of the likelihood of the training data given the parameters, i.e.

$$\mathcal{P}_\mathcal{G}^{\mathrm{ML}} = \arg\max_{\mathcal{P}_\mathcal{G}} \prod_{n=1}^{N} \mathrm{P}^\mathcal{B}(c^{(n)}, \mathbf{x}^{(n)}). \tag{4}$$

  Note that the above optimization problem implicitly includes sum-to-one constraints because the learned parameters in $\mathcal{P}_\mathcal{G}^{\mathrm{ML}}$ must represent normalized probabilities. Maximum likelihood parameters minimize the Kullback-Leibler (KL)-divergence between $\mathrm{P}^\mathcal{B}(C, \mathbf{X})$ and $\mathrm{P}^*(C, \mathbf{X})$ [7].
- **Discriminative Maximum Margin Parameters [5,12,15].** In discriminative learning one aims at identifying parameters leading to good classification performance on new samples from $\mathrm{P}^*(C, \mathbf{X})$. This type of learning is for example advantageous in cases where the assumed model distribution $\mathrm{P}^\mathcal{B}(C, \mathbf{X})$ cannot approximate $\mathrm{P}^*(C, \mathbf{X})$ well, for example because of a too limited BN structure [17].
  Discriminative MM parameters $\mathcal{P}_\mathcal{G}^{\mathrm{MM}}$ are found as

$$\mathcal{P}_\mathcal{G}^{\mathrm{MM}} = \arg\max_{\mathcal{P}_\mathcal{G}} \prod_{n=1}^{N} \min\left(\gamma, d^\mathcal{B}(c^{(n)}, \mathbf{x}^{(n)})\right), \tag{5}$$

[5] Note that the parameter learning approach can be applied to more complex structures, e.g. k-trees among the attributes.

where $d^{\mathcal{B}}(c^{(n)}, \mathbf{x}^{(n)})$ is the margin of the n^{th} sample given as

$$d^{\mathcal{B}}(c^{(n)}, \mathbf{x}^{(n)}) = \frac{\mathrm{P}^{\mathcal{B}}(c^{(n)}, \mathbf{x}^{(n)})}{\max_{c \neq c^{(n)}} \mathrm{P}^{\mathcal{B}}(c, \mathbf{x}^{(n)})}, \tag{6}$$

and where the hinge loss function is denoted as $\min(\gamma, d^{\mathcal{B}}(c^{(n)}, \mathbf{x}^{(n)}))$. The parameter $\gamma > 1$ controls the margin. In this way, the margin *measures* the ratio of the likelihood of the n^{th} sample belonging to the correct class $c^{(n)}$ to the likelihood of belonging to the most likely competing class. The n^{th} sample is correctly classified iff $d^{\mathcal{B}}(c^{(n)}, \mathbf{x}^{(n)}) > 1$ and vice versa.

4 Algorithms for Online Learning of Reduced-Precision Parameters

We start by considering learning ML parameters in Section 4.1 and then move on to learning MM parameters in Section 4.2. We claim that learning using reduced-precision arithmetic is most useful in online settings, i.e. parameters are updated on a per-sample basis. This online learning scenario captures the important case in which initially pre-computed parameters are used and these parameters are updated online as new samples become available, e.g. adaptation of a hearing-aid to a new acoustic environment. In this setting, learning using reduced-precision computations requires specialized algorithms, i.e. gradient-descent (or gradient-ascent) procedures using reduced-precision arithmetic do not perform well. The reason is that the necessary exact projections of the parameters onto the sum-to-one constraints cannot be accurately performed. Another issue is the limited resolution of the learning rate. However, we find this issue less important as the inexact projections.

4.1 Learning Maximum Likelihood Parameters

We consider an online algorithm for learning ML parameters. The ML objective (4) for the offline scenario can be equivalently written as

$$\mathbf{w}^{\mathrm{ML}} = \arg\max_{\mathbf{w}} \sum_{n=1}^{N} \boldsymbol{\phi}(c^{(n)}, \mathbf{x}^{(n)})^T \mathbf{w} \quad \text{s.t.} \sum_{j} \exp(w^i_{j|\mathbf{h}}) = 1, \forall i, j, \mathbf{h}, \tag{7}$$

where optimisation is performed over the log-parameters $\mathbf{w}$. In an online scenario, not all samples are available for learning at once but are available one at a time; the parameters $\mathbf{w}^{\mathrm{ML},t}$ at time-step t are updated according to the gradient of a single sample $(c, \mathbf{x})$ (or, alternatively, a batch of samples) and projected such that they satisfy the sum-to-one constraints, i.e.

$$\mathbf{w}^{\mathrm{ML},t+1} = \Pi\left[\mathbf{w}^{\mathrm{ML},t} + \eta\left(\nabla_{\mathbf{w}} \boldsymbol{\phi}(c, \mathbf{x})^T \mathbf{w}\right)(\mathbf{w}^{\mathrm{ML},t})\right] \tag{8}$$

$$= \Pi\left[\mathbf{w}^{\mathrm{ML},t} + \eta \boldsymbol{\phi}(c, \mathbf{x})\right], \tag{9}$$

where η is the learning rate, $\nabla_{\mathbf{w}}(f)(\mathbf{a})$ denotes the gradient of f with respect to $\mathbf{w}$ at $\mathbf{a}$, and $\Pi[\mathbf{w}]$ denotes the ℓ_2-norm projection of the parameter vector $\mathbf{w}$ onto the set of normalized parameter vectors. Note that the gradient has a simple form: it consists only of zeros and ones, where the ones are *indicators* of *active* entries in the CPTs of sample $(c, \mathbf{x})$. Furthermore, assuming normalized parameters at time-step t, the direction of the gradient is always such that the parameters $\mathbf{w}^{\mathrm{ML},t+1}$ are super-normalized. Consequently, after (exact) projection the parameters satisfy the sum-to-one constraints.

We continue by analyzing the effect of using reduced-precision arithmetic on the online learning algorithm. Therefore, we performed the following experiment: Assume that the projection can only be approximately performed. We *simulate* the approximate projection by performing an exact projection and subsequently adding quantization noise (this is similar to reduced-precision analysis in signal processing [10]). We sample the noise from a Gaussian distribution with zero mean and with variance $\sigma^2 = q^2/12$, where $q = 2^{-b_f}$. For the satimage dataset from the UCI repository [1] we construct BNCs with TAN structure. As initial parameters we use rounded ML parameters computed from one tenth of the training data. Then, we present the classifier further samples in an online manner and update the parameters according to (9). During learning, we set the learning rate η to $\eta = \eta_0/\sqrt{1+t}$, where η_0 is some constant (η_0 is tuned by hand such that the test set performance is maximized). The resulting classification performance is shown in Figures 2a and 2b for the exact and the approximate projection, respectively. One can observe, that the algorithm does not properly learn using the approximate projection. Thus, it seems crucial to perform the projections rather accurately. To circumvent the need for accurate projections, we propose a method that avoids computing a projection at all in the following.

Consider again the offline parameter learning case. ML parameters can be computed in closed-form by computing relative frequencies, i.e.

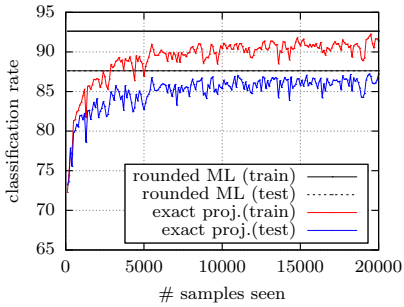
$$\theta^i_{j|\mathbf{h}} = \frac{m^i_{j|\mathbf{h}}}{m^i_{\mathbf{h}}}, \tag{10}$$

where

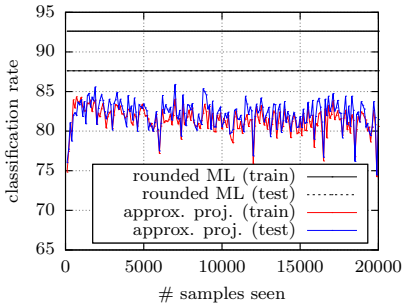
$$m^i_{j|\mathbf{h}} = \sum_{n=1}^{N} \boldsymbol{\phi}(c^{(n)}, \mathbf{x}^{(n)})^i_{j|\mathbf{h}}, \text{ and } m^i_{\mathbf{h}} = \sum_j m^i_{j|\mathbf{h}}. \tag{11}$$

This can be easily extended to online learning. Assume that the counts $m^{i,t}_{j|\mathbf{h}}$ at time t are given and that a sample $(c^t, \mathbf{x}^t)$ is presented to the learning algorithm. Then, the counts are updated according to

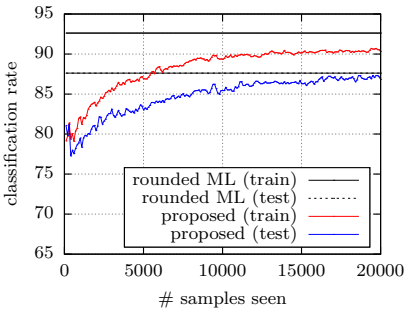
$$m^{i,t+1}_{j|\mathbf{h}} = m^{i,t}_{j|\mathbf{h}} + \boldsymbol{\phi}(c^t, \mathbf{x}^t)^i_{j|\mathbf{h}}. \tag{12}$$



(a) exact projection



(b) approximate projection



(c) proposed algorithm

Fig. 2. Classification performance of BNCs with TAN structure for satimage data in an online learning scenario; (a) Online ML parameter learning with exact projection after each parameter update, (b) online ML parameter learning with approximate projection after each parameter update (see text for details), (c) proposed algorithm for online ML parameter learning.

Exploiting these counts, the logarithm of the ML parameters $\theta^{i,t}_{j|\mathbf{h}}$ at time t can be computed as

$$w^{i,t}_{j|\mathbf{h}} = \log\left(\frac{m^{i,t}_{j|\mathbf{h}}}{m^{i,t}_{\mathbf{h}}}\right), \tag{13}$$

where similarly to before $m^{i,t}_{\mathbf{h}} = \sum_j m^{i,t}_{j|\mathbf{h}}$. A straightforward approximation of (13) is to (approximately) compute the counts $m^{i,t}_{j|\mathbf{h}}$ and $m^{i,t}_{\mathbf{h}}$, respectively, and to use a lookup table to determine $w^{i,t}_{j|\mathbf{h}}$. The lookup table can be indexed in terms of $m^{i,t}_{j|\mathbf{h}}$ and $m^{i,t}_{\mathbf{h}}$ and stores values for $w^{i,t}_{j|\mathbf{h}}$ in the desired reduced-precision format. To limit the maximum size of the lookup table and the bit-width required for the counters for $m^{i,t}_{j|\mathbf{h}}$ and $m^{i,t}_{\mathbf{h}}$, we assume some maximum integer number M. We pre-compute the lookup table L such that

$$L(i,j) = \left[\frac{\log_2(i/j)}{q}\right]_R \cdot q, \tag{14}$$

where $[\cdot]_R$ denotes rounding to the closest the integer, q is the quantization interval of the desired fixed-point representation, $\log_2(\cdot)$ denotes the base-2 logarithm, and where i and j are in the range $0, \ldots, M-1$. Given sample $(c^t, \mathbf{x}^t)$, the counts $m_{j|\mathbf{h}}^{i,t+1}$ and $m_{\mathbf{h}}^{i,t+1}$ are computed according to Algorithm 1 from the counts $m_{j|\mathbf{h}}^{i,t}$ and $m_{\mathbf{h}}^{i,t}$. To guarantee that the counts stay in range, the algorithm identifies counters that reach their maximum value, and halfs these counters as well as all other counters corresponding to the same CPTs. This division by 2 can be implemented as a bitwise shift operation.

Algorithm 1. Reduced-precision ML online learning

Require: Old counts $m_{j|\mathbf{h}}^{i,t}$; sample $(c^t, \mathbf{x}^t)$
  $m_{j|\mathbf{h}}^{i,t+1} \leftarrow m_{j|\mathbf{h}}^{i,t} + \boldsymbol{\phi}(c^t, \mathbf{x}^t)_{j|\mathbf{h}}^{i} \quad \forall i, j, \mathbf{h}$ ▷ update counts
  for $i, j, \mathbf{h}$ **do**
    if $m_{j|\mathbf{h}}^{i,t+1} = M$ **then** ▷ maximum value of counter reached?
      $m_{j|\mathbf{h}}^{i,t+1} \leftarrow \lfloor m_{j|\mathbf{h}}^{i,t+1}/2 \rfloor \quad \forall j$ ▷ half counters of considered CPT (round down)
    end if
  end for
  return $m_{j|\mathbf{h}}^{i,t+1}$

Initially, we set all counts to zero, i.e. $m_{j|\mathbf{h}}^{i,0} = 0$, respectively. For the cumulative counts, i.e. $m_{\mathbf{h}}^{i,t}$ in (13), we did not limit the number of bits (for real implementations the necessary number of bits for this counter can be computed from the bit-width of the individual counters that are summed up and the graph structure of the considered BNC). Logarithmic parameters $w_{j|\mathbf{h}}^{i,t}$ are computed using the lookup table described above and using Algorithm 2. The classification performance during online learning is shown in Figure 2c. We can observe, that the algorithm behaves pleasant and the limited range of the used counters does not seem to affect classification performance (compared to the classification performance using rounded ML parameters computed using full-precision computations and all training samples). Further experimental results can be found in Section 5.

4.2 Learning Maximum Margin Parameters

In this section, we consider a variant of the MM objective proposed in [12] that balances the MM objective (5) against the ML objective (4), i.e. the objective is to maximize

Algorithm 2. Computation of logarithmic probabilities from lookup table

Require: Counts $m_{j|\mathbf{h}}^{i,t}$ and $m_{\mathbf{h}}^{i,t}$; lookup table L of size $M \times M$
  div $\leftarrow 0$
  while $m_{\mathbf{h}}^{i,t} \geq M$ **do** ▷ ensure that index into lookup table is in range
    $m_{\mathbf{h}}^{i,t} \leftarrow \lfloor m_{\mathbf{h}}^{i,t}/2 \rfloor$ ▷ half and round down
    div $\leftarrow$ div $+ 1$
  end while
  $w_{j|\mathbf{h}}^{i,t} \leftarrow L(m_{j,\mathbf{h}}^{i,t}, m_{\mathbf{h}}^{i,t}) \quad \forall j$ ▷ get log-probability from lookup table
  while div > 0 and $\forall j \colon w_{j|\mathbf{h}}^{i,t} > (-2^{b_i} + 2^{b_f}) + 1$ **do** ▷ revise index correction
    $w_{j|\mathbf{h}}^{i,t} \leftarrow w_{j|\mathbf{h}}^{i,t} - 1 \quad \forall j$
    div $\leftarrow$ div $- 1$
  end while
  return $w_{j|\mathbf{h}}^{i,t}$

$$\underbrace{\log\left[\prod_{n=1}^{N} \mathrm{P}^{\mathcal{B}}(c^{(n)}, \mathbf{x}^{(n)})\right]}_{\text{ML}} + \lambda \underbrace{\log\left[\prod_{n=1}^{N} \min\left(\gamma, \frac{\mathrm{P}^{\mathcal{B}}(c^{(n)}, \mathbf{x}^{(n)})}{\max_{c \neq c^{(n)}} \mathrm{P}^{\mathcal{B}}(c, \mathbf{x}^{(n)})}\right)\right]}_{\text{MM}}. \tag{15}$$

In this way, generative properties, e.g. the ability to marginalize over missing features, are combined with good discriminative performance. This variant of the MM objective can be easily written in the form

$$\mathbf{w}^{\mathrm{MM}} = \arg\max_{\mathbf{w}} \left[\sum_{n=1}^{N} \boldsymbol{\phi}(c^{(n)}, \mathbf{x}^{(n)})^T \mathbf{w} + \lambda \sum_{n=1}^{N} \min\left(\gamma, \min_{c \neq c^{(n)}} \left[(\boldsymbol{\phi}(c^{(n)}, \mathbf{x}^{(n)}) - \boldsymbol{\phi}(c, \mathbf{x}^{(n)}))^T \mathbf{w}\right]\right)\right] \tag{16}$$

and, for simplicity, we will refer to this modified objective as the MM objective. Note that there are implicit sum-to-one constraints in problem (16), i.e. any feasible solution $\mathbf{w}$ must satisfy $\sum_j \exp(w_{j|\mathbf{h}}^i) = 1$ for all $i, j, \mathbf{h}$. In the online learning case, given sample $(c, \mathbf{x})$, the parameters $\mathbf{w}^{\mathrm{MM},t+1}$ at time $t+1$ are computed from the parameters $\mathbf{w}^{\mathrm{MM},t}$ at time t as

$$\mathbf{w}^{\mathrm{MM},t+1} = \mathit{\Pi}\left[\mathbf{w}^{\mathrm{MM},t} + \eta \boldsymbol{\phi}(c, \mathbf{x}) + \eta\lambda \mathbf{g}(c, \mathbf{x})\right], \tag{17}$$

where

$$\mathbf{g}(c, \mathbf{x}) = \begin{cases} 0 & \min_{c' \neq c} \left[(\boldsymbol{\phi}(c, \mathbf{x}) - \boldsymbol{\phi}(c', \mathbf{x}))^T \mathbf{w}\right] \geq \gamma, \\ \boldsymbol{\phi}(c, \mathbf{x}) - \boldsymbol{\phi}(c', \mathbf{x}) & \text{o.w., } c' = \arg\min_{c'} \left[(\boldsymbol{\phi}(c, \mathbf{x}) - \boldsymbol{\phi}(c', \mathbf{x}))^T \mathbf{w}\right] \end{cases} \tag{18}$$

and where similar as before $\mathit{\Pi}[\mathbf{w}]$ denotes the projection.

For learning MM parameters, a similar observation with respect to the accuracy of the projection can be made as for ML parameters. But we cannot proceed exactly as in the case of learning ML parameters because we cannot compute MM parameters in closed-form. As in the ML parameter learning case, the gradient for the parameter update has a rather simple form, but the projection to satisfy the sum-to-one constraints is difficult to compute. Therefore, for online MM parameter learning, we propose Algorithm 3 that is similar to Algorithm 1 in Section 4.1, i.e. we avoid to compute the projection explicitly. From the counts computed by the algorithm, log-probabilities can be computed using Algorithm 2. Note that the proposed algorithm does not exactly optimize (16) but a, not explicitly defined, surrogate. The idea behind the algorithm is (1) to optimize the likelihood term in (16) as in the algorithm for ML parameter learning, and (2) to optimize the margin term by increasing the likelihood for the correct class and simultaneously decreasing the likelihood for the strongest competitor class. Note that the idea of optimizing the margin term as explained above is similar in spirit to that of discriminative frequency estimates [19]. However, discriminative frequency estimates do not optimize a margin term but a term more closely related to the class-conditional likelihood.

5 Experiments

5.1 Datasets

In our experiments, we considered the following datasets.

1. **UCI data [1].** This is in fact a large collection of datasets, with small to medium number of samples. Features are discretized as needed using the algorithm proposed in [3]. If not stated otherwise, in case of the datasets *chess, letter, mofn-3-7-10, segment, shuttle-small, waveform-21, abalone, adult, car, mushroom, nursery, and spambase*, a test set was used to estimate the accuracy of the classifiers. For all other datasets, classification accuracy was estimated by 5-fold cross-validation. Information on the number of samples, classes and features for each dataset can be found in [1].

2. **USPS data [6].** This data set contains 11000 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 greyscale image. These greyscale values are discriminatively quantized [3] and each pixel is considered as feature.

3. **MNIST Data [8].** This dataset contains 70000 samples of handwritten digits. In the standard setting, 60000 samples are used for training and 10000 for testing. The digits represented by grey-level images were down-sampled by a factor of two resulting in a resolution of 16×16 pixels, i.e. 196 features.

5.2 Results

We performed experiments using $M = 1024$, i.e. we used counters with 10 bits ($b_i + b_f = 10$). The splitting of the available bits into integer bits and fractional bits was set using 10-fold cross-validation. Experimental results for BNCs

Algorithm 3. Reduced-precision MM online learning

Require: Old counts $m_{j|\mathbf{h}}^{i,t}$; sample $(c^t, \mathbf{x}^t)$; hyper-parameters $\gamma, \lambda \in \mathbb{N}_+$ for MM formulation

$m_{j|\mathbf{h}}^{i,t+1} \leftarrow m_{j|\mathbf{h}}^{i,t} + \boldsymbol{\phi}(c^t, \mathbf{x}^t)_{j|\mathbf{h}}^{i} \quad \forall i, j, \mathbf{h}$ ▷ update counts (likelihood term)

for $i, j, \mathbf{h}$ **do** ▷ ensure that parameters stay in range

  if $m_{j|\mathbf{h}}^{i,t+1} = M$ **then**

    $m_{j|\mathbf{h}}^{i,t+1} \leftarrow \lfloor m_{j|\mathbf{h}}^{i,t+1}/2 \rfloor \quad \forall j$

  end if

end for

$c' \leftarrow$ strongest competitor of class c for features $\mathbf{x}$

if $\left[(\boldsymbol{\phi}(c^t, \mathbf{x}^{(n)}) - \boldsymbol{\phi}(c', \mathbf{x}^{(n)}))^T \mathbf{w} < \gamma \right]$ **then**

  $m_{j|\mathbf{h}}^{i,t+1} \leftarrow m_{j|\mathbf{h}}^{i,t} \quad \forall i, j, \mathbf{h}$

  for $k = 1, \ldots, \lambda$ **do** ▷ Add-up gradient in λ steps

    $m_{j|\mathbf{h}}^{i,t+1} \leftarrow m_{j|\mathbf{h}}^{i,t+1} + \boldsymbol{\phi}(c^t, \mathbf{x}^t)_{j|\mathbf{h}}^{i} \quad \forall i, j, \mathbf{h}$ ▷ update counts (margin term)

    $m_{j|\mathbf{h}}^{i,t+1} \leftarrow m_{j|\mathbf{h}}^{i,t+1} - \boldsymbol{\phi}(c', \mathbf{x}^t)_{j|\mathbf{h}}^{i} \quad \forall i, j, \mathbf{h}$ ▷ update counts (margin term)

    for $i, j, \mathbf{h}$ **do** ▷ ensure that parameters stay in range

      if $m_{j|\mathbf{h}}^{i,t+1} = 0$ **then**

        $m_{j|\mathbf{h}}^{i,t+1} \leftarrow m_{j|\mathbf{h}}^{i,t+1} + 1 \quad \forall j$

      end if

      if $m_{j|\mathbf{h}}^{i,t+1} = M$ **then**

        $m_{j|\mathbf{h}}^{i,t+1} \leftarrow \lfloor m_{j|\mathbf{h}}^{i,t+1}/2 \rfloor \quad \forall j$

      end if

    end for

  end for

end if

return $m_{j|\mathbf{h}}^{i,t+1}$

with NB and TAN structures are shown in Table 1 for the datasets described above. All samples from the training set were presented to the proposed algorithm twenty times in random order. The absolute reduction in classification rate (CR) compared to the *exact* CR, i.e. using BNCs with the optimal double-precision parameters, for the considered datasets is, with few exceptions, relatively small. Thus the proposed reduced-precision computation scheme seems to be sufficiently accurate to yield good classification performance while employing only range-limited counters and a lookup table of size $M \times M$. Clearly, the performance of the proposed method can be improved by using larger and more accurate lookup tables and counters with larger bit-width.

For discriminative parameter learning, we set the hyper-parameters $\lambda \in \{0, 1, 2, 4, 8, 16\}$ and $\gamma \in \{0.25, 0.5, 1, 2, 4, 8.\}$ using 10-fold cross-validation. For this setup, we observed the classification performance summarized in Table 1. While the results are not as good as those of the exact MM solution, in terms of the absolute reduction in CR, we can clearly observe an improvement in classification performance using the proposed MM parameter learning method over the proposed ML parameter learning method for many datasets. The performance of BNCs using

Table 1. Classification performance. CRs using ML/MM parameters according to (10)/(16) in double-precision are denoted as *ML exact*/*MM exact*. CRs using reduced-precision ML/MM parameters computed according to Algorithm 1/Algorithm 3 using only reduced-precision arithmetic are denoted as *ML prop.*/*MM prop.*; *ML abs.*/*MM abs.* denote the absolute reduction in CR for double-precision ML/MM parameters to reduced-precision ML/MM parameters.

		ML − CR [%]			MM − CR [%]		
Dataset	Structure	exact	prop.	abs.	exact	prop.	abs.
USPS	NB	86.89	86.34	0.55	93.91	93.17	0.74
	TAN	91.39	90.05	1.34	93.01	93.50	−0.49
MNIST	NB	82.88	80.61	2.26	93.11	93.00	0.11
	TAN	90.49	87.92	2.57	93.49	93.83	−0.34
australian	NB	85.92	85.48	0.44	87.24	85.63	1.61
	TAN	81.97	84.46	−2.49	84.76	83.58	1.18
breast	NB	97.63	97.48	0.15	97.04	97.63	−0.59
	TAN	95.85	96.15	−0.30	96.00	94.52	1.48
chess	NB	87.45	86.20	1.25	97.68	94.32	3.36
	TAN	92.19	92.13	0.06	97.99	96.27	1.73
cleve	NB	82.87	83.55	−0.68	82.53	80.84	1.69
	TAN	79.09	80.47	−1.37	80.79	75.69	5.10
corral	NB	89.16	89.22	−0.07	93.36	93.36	0.00
	TAN	97.53	94.96	2.57	100.00	99.20	0.80
crx	NB	86.84	86.22	0.62	86.06	86.68	−0.62
	TAN	83.73	84.04	−0.31	84.20	83.58	0.62
diabetes	NB	73.96	72.65	1.31	74.87	75.01	−0.14
	TAN	73.83	73.44	0.39	74.35	71.73	2.62
flare	NB	77.16	75.81	1.34	83.11	83.97	−0.86
	TAN	83.59	79.46	4.13	83.30	83.20	0.10
german	NB	74.50	72.90	1.60	75.30	73.80	1.50
	TAN	72.60	71.80	0.80	72.60	72.10	0.50
glass	NB	71.16	71.66	−0.50	70.61	71.08	−0.47
	TAN	71.11	69.58	1.53	72.61	69.55	3.05
heart	NB	81.85	82.96	−1.11	83.33	84.44	−1.11
	TAN	81.48	81.11	0.37	81.48	81.48	0.00
hepatitis	NB	89.83	89.83	0.00	92.33	88.67	3.67
	TAN	84.83	87.33	−2.50	86.17	88.58	−2.42
letter	NB	74.95	74.41	0.54	85.79	81.50	4.30
	TAN	86.26	85.93	0.33	88.57	88.43	0.14
lymphography	NB	84.23	85.71	−1.48	82.80	87.31	−4.51
	TAN	82.20	82.86	−0.66	76.92	80.66	−3.74
nursery	NB	89.97	89.63	0.35	93.03	93.05	−0.02
	TAN	92.87	92.87	0.00	98.68	98.12	0.56
satimage	NB	81.56	82.02	−0.45	88.41	86.96	1.45
	TAN	85.85	86.40	−0.55	86.98	87.44	−0.47
segment	NB	92.68	91.90	0.78	95.37	93.85	1.52
	TAN	94.85	94.89	−0.04	95.76	95.63	0.13
shuttle	NB	99.66	99.10	0.56	99.95	99.86	0.09
	TAN	99.88	99.71	0.17	99.93	99.87	0.06
soybean-large	NB	93.35	92.80	0.56	91.50	92.05	−0.55
	TAN	91.14	89.12	2.02	91.87	92.61	−0.74
spambase	NB	90.03	89.88	0.15	94.08	93.19	0.89
	TAN	92.97	92.79	0.17	94.03	93.73	0.31
vehicle	NB	61.57	61.93	−0.36	67.95	69.16	−1.21
	TAN	71.09	68.91	2.18	69.88	69.16	0.72
vote	NB	90.16	90.63	−0.47	94.61	94.61	0.00
	TAN	94.61	94.60	0.01	95.31	94.60	0.71
waveform-21	NB	81.14	81.18	−0.04	85.14	84.16	0.98
	TAN	82.52	82.20	0.32	83.48	83.94	−0.46

the optimal double-precision parameters is in many cases not significantly better. Note that the hyper-parameters used for determining double-precision parameters are different than those used for determining reduced-precision parameters, i.e. a larger range of values is used (details are provided in [12]). The larger range of values cannot be used in case of reduced-precision parameters because of the limited parameter resolution.

6 Discussions

We proposed online algorithms for learning BNCs with reduced-precision fixed-point parameters using reduced-precision computations only. This facilitates the utilization of BNCs in computationally constrained platforms, e.g. embedded- and ambient-systems, as well as power-aware systems. The algorithms differ from naive implementations of conventional algorithms by avoiding error-prone parameter projections commonly used in gradient ascent/descent algorithms. In experiments, we demonstrated that our algorithms yield parameters that achieve classification performances close to that of optimal double-precision parameters for many of the investigated datasets.

Our algorithms have similarities with a very simple method for learning discriminative parameters of BNCs known as *discriminative frequency estimates* [19]. According to this method, parameters are estimated using a perceptron-like algorithm, where parameters are updated by the prediction loss, i.e. the difference of the class posterior of the correct class (which is assumed to be 1 for the data in the training set) and the class posterior according to the model using the current parameters.

References

1. Bache, K., Lichman, M.: UCI Machine Learning Repository (2013). http://archive.ics.uci.edu/ml
2. Courbariaux, M., Bengio, Y., David, J.: Low precision arithmetic for deep learning. CoRR abs/1412.7024 (2014). http://arxiv.org/abs/1412.7024
3. Fayyad, U.M., Irani, K.B.: Multi-Interval discretization of continuous-valued attributes for classification learning. In: International Conference on Artificial Intelligence (IJCAI), pp. 1022–1029 (2003)
4. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian Network Classifiers. Machine Learning **29**, 131–163 (1997)
5. Guo, Y., Wilkinson, D., Schuurmans, D.: Maximum Margin Bayesian Networks. In: Uncertainty in Artificial Intelligence (UAI), pp. 233–242 (2005)
6. Hull, J.J.: A Database for Handwritten Text Recognition Research. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **16**(5), 550–554 (1994)
7. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
8. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based Learning Applied to Document Recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

9. Lee, D.U., Gaffar, A.A., Cheung, R.C.C., Mencer, O., Luk, W., Constantinides, G.A.: Accuracy-Guaranteed Bit-Width Optimization. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **25**(10), 1990–2000 (2006)
10. Oppenheim, A.V., Schafer, R.W., Buck, J.R.: Discrete-time Signal Processing, 2nd edn., Prentice-Hall Inc. (1999)
11. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc. (1988)
12. Peharz, R., Tschiatschek, S., Pernkopf, F.: The most generative maximum margin bayesian networks. In: International Conference on Machine Learning (ICML), vol. 28, pp. 235–243 (2013)
13. Pernkopf, F., Peharz, R., Tschiatschek, S.: Introduction to Probabilistic Graphical Models, vol. 1, chap. 18, pp. 989–1064. Elsevier (2014)
14. Pernkopf, F., Wohlmayr, M., Mücke, M.: Maximum margin structure learning of bayesian network classifiers. In: International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2076–2079 (2011)
15. Pernkopf, F., Wohlmayr, M., Tschiatschek, S.: Maximum Margin Bayesian Network Classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **34**(3), 521–531 (2012)
16. Piatkowski, N., Sangkyun, L., Morik, K.: The integer approximation of undirected graphical models. In: International Conference on Pattern Recognition Applications and Methods (ICPRAM) (2014)
17. Roos, T., Wettig, H., Grünwald, P., Myllymäki, P., Tirri, H.: On Discriminative Bayesian Network Classifiers and Logistic Regression. Journal of Machine Learning Research **59**(3), 267–296 (2005)
18. Soudry, D., Hubara, I., Meir, R.: Expectation backpropagation: parameter-free training of multilayer neural networks with continuous or discrete weights. In: Advances in Neural Information Processing Systems, pp. 963–971 (2014)
19. Su, J., Zhang, H., Ling, C.X., Matwin, S.: Discriminative parameter learning for bayesian networks. In: International Conference on Machine Learning (ICML), pp. 1016–1023. ACM (2008)
20. Tong, J.Y.F., Nagle, D., Rutenbar, R.A.: Reducing Power by Optimizing the Necessary Precision/Range of Floating-point Arithmetic. IEEE Transactions on Very Large Scale Integration Systems **8**(3), 273–285 (2000)
21. Tschiatschek, S., Cancino Chacón, C.E., Pernkopf, F.: Bounds for bayesian network classifiers with reduced precision parameters. In: International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3357–3361 (2013)
22. Tschiatschek, S., Paul, K., Pernkopf, F.: Integer bayesian network classifiers. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.) ECML PKDD 2014, Part III. LNCS, vol. 8726, pp. 209–224. Springer, Heidelberg (2014)
23. Tschiatschek, S., Pernkopf, F.: On Reduced Precision Bayesian Network Classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (to be published)
24. Tschiatschek, S., Reinprecht, P., Mücke, M., Pernkopf, F.: Bayesian network classifiers with reduced precision parameters. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part I. LNCS, vol. 7523, pp. 74–89. Springer, Heidelberg (2012)