The Integer Approximation of Undirected Graphical Models

Nico Piatkowski, Sangkyun Lee and Katharina Morik
Artificial Intelligence Group, TU Dortmund University, 44227 Dortmund, Germany

Keywords: Graphical Models, Approximate Inference.

Abstract: Machine learning on resource constrained ubiquitous devices suffers from high energy consumption and slow execution time. In this paper, it is investigated how to modify machine learning algorithms in order to reduce the number of consumed clock cycles—not by reducing the asymptotic complexity, but by assuming a weaker execution platform. In particular, an integer approximation to the class of undirected graphical models is proposed. Algorithms for inference, maximum-a-posteriori prediction and parameter estimation are presented and approximation error is discussed. In numerical evaluations on synthetic data, the response of the model to several influential properties of the data is investigated. The results on the synthetic data are confirmed with a natural language processing task on an open data set. In addition, the runtime on low-end hardware is regarded. The overall speedup of the new algorithms is at least $2\times$ while overall loss in accuracy is rather small. This allows running probabilistic methods on very small devices, even if they do not contain a processor that is capable of executing floating point arithmetic at all.

1 INTRODUCTION

Data analytics for streaming sensor data brings challenges for the resource efficiency of algorithms in terms of execution time and the energy consumption simultaneously. Fortunately, optimizations which reduce the number of CPU cycles also reduce energy consumption. When reviewing the specifications of processing units, one finds that integer arithmetic is usually cheaper in terms of instruction latency, i.e. it needs a small number of clock cycles until the result of an arithmetic instruction is ready. This motivates the reduction of CPU cycles in which code is executed when designing a new, resource-aware learning algorithm. Beside clock cycle reduction, limited memory usage is also an important factor for small devices.

Outsourcing parts of data analysis from data centers to ubiquitous devices that actually measuredata would reduce the communication costs and thus energy consumption. If, for instance, a mobile medical device or smartphone can build a probabilistic model of the usage behavior of its user, energy models can be made more accurate and power management can be more efficient. The biggest hurdle in doing this, are the heavily restricted computational capabilities of very small devices—some do not even have a floating point processor. Consequently, computationally simple machine learning approaches have to be considered. Low complexity of machine learning models is usually achieved by independence assumptions among features or labels. In contrast, the joint prediction of multiple dependent variables based on multiple observed inputs is an ubiquitous subtask in real world problems from various domains. Probabilistic graphical models are well suited for such tasks, but they suffer from the high complexity of probabilistic inference.

In the paper at hand, it is shown that the framework of undirected graphical models (Wainwright and Jordan, 2007) can be mapped to an integer domain. Inference algorithms and a new optimization scheme are proposed, that allow the learning of integer parameters without the need for any floating point computation. This opens up the opportunity of running machine learning tasks on very small, resource-constrained devices. To be more precise, based only on integers, it is possible to compute approximations to marginal probabilities, to maximum-a-posteriori (MAP) assignments and maximum likelihood estimate either via an approximate closed form solution or an integer variant of the stochastic gradient descent (SGD) algorithm. It turns out that the integer approximations use less memory and deliver a reasonable quality while being around twice as fast as their floating point counterparts. To the best of our knowledge, there is nothing like an integer undirected model so far. The remainder of this paper is organized as follows. This Section continues with an overview

Piatkowski N., Lee S. and Morik K..
The Integer Approximation of Undirected Graphical Models.
DOI: 10.5220/0004831202960304
In *Proceedings of the 3rd International Conference on Pattern Recognition Applications and Methods* (ICPRAM-2014), pages 296-304
ISBN: 978-989-758-018-5

on related work. A short introduction to probabilistic graphical models is given in Section 2. In Section 3, the intuition behind integer undirected graphical models is explained, and the corresponding algorithms are derived. Furthermore, a bound on the training error is presented. Two instances of the integer framework, *Integer Markov Random Fields* and *Integer Conditional Random Fields*, are evaluated in Section 4 on synthetic and real world data. Finally, Section 5 concludes this work.

1.1 Related Work

Many approximate approaches to probabilistic inference based on Belief Propagation (BP) (Kschischang et al., 2001; Pearl, 1988) were proposed in the last decade. Among them Counting BP (Kersting et al., 2009), Lifted BP (Ahmadi et al., 2012), Stochastic BP (Noorshams and Wainwright, 2011), Tree-reweighted BP (Wainwright et al., 2003), Tree Block Coordinate Descent (Sontag and Jaakkola, 2009) or Particle BP (Ihler and McAllester, 2009). Unfortunately, most of these methods are by no means suited for embedded or resource constraint environments. In contrast to these approaches, the model class that is proposed in the paper at hand has the same asymptotic complexity as the vanilla inference methods, but it uses cheaper operations. Inspired by work from the signal processing community (Hassibi and Boyd, 1998), the underlying model class is restricted to the integers, which results in a reduced runtime and energy savings, while keeping a good performance. This new approach should not be confused with models that are designed for integer state spaces, in which case the state space $\mathcal{X}$ is a subset of the natural numbers or, more generally, is a metric space. Here, the state space may be an arbitrary discrete space without any additional constraints.

Estimation in discrete parameter models was recently investigated in (Choirat and Seri, 2012). They discuss consistency, asymptotic distribution theory, information inequalities and their relations with efficiency and super-efficiency for a general class of *m*-estimators. Unfortunately, the authors do not consider the case when the true estimator is not included in the search space and therefore, their analysis cannot be used to estimate the error in a situation when the optimizer has to be approximated.

Bayesian network classifiers with reduced precision parameters have been introduced recently (Tschiatschek et al., 2012). The authors evaluate empirically the classification performance when reducing the floating-point precision of probability parameters of Bayesian networks. After learning the parameters as usual in $\mathbb{R}$ (represented as 64 bit double precision floating point numbers), they varied the bit-width of mantissa and exponent, and reported the prediction accuracy in terms of the normalized number of correctly classified test instances. They found that after learning, the parameters may be multiplied by a sufficiently large integer constant (10^9) to convert the probabilities into integer numbers. Therefore, their method still relies on floating point arithmetic for learning and prediction. However, Tschiatschek et al. missed the point that real valued probability parameters are necessary for Bayesian networks but not for all classes of probabilistic graphical models.

2 PROBABILISTIC GRAPHICAL MODELS

The basic notation and concepts of probabilistic graphical models in this Section are based on (Wainwright and Jordan, 2007), which is an excellent introduction to this topic. Let $G = (V,E)$ be an undirected graph with $|V| = n$ vertices, edge set $E \subset V \times V$ and $\mathcal{N}_v := \{w \in V : \{v,w\} \in E\}$ the neighbors of vertex $v \in V$. Each vertex $v \in V$ corresponds to a random variable X_v with a realization x_v and a domain $\mathcal{X}_v$ with at least two different states, i.e. $|\mathcal{X}_v| \geq 2$. Consider an n-dimensional random variable $X = (X_v)_{v \in V}$ with realization $x \in \mathcal{X} = \otimes_{v \in V} \mathcal{X}_v$. The probability of the event $\{X = x\}$ is denoted by $p(X = x)$. $p(x)$ is used as a shortcut for $p(X = x)$ in the remainder . For a set of vertices $A \subseteq V$, X_A is addressing the components of X that correspond to the vertices in A. For ease of notation, X_v and $X_{\{v\}}$ are regarded the same. For undirected graphical models, the joint probability mass function of X is given by

$$p_\theta(x) = \frac{1}{Z(\theta)} \prod_{C \in \mathcal{C}(G)} \psi_C(x_C) \tag{1}$$

$$Z(\theta) = \sum_{x \in \mathcal{X}} \prod_{C \in \mathcal{C}(G)} \psi_C(x_C) \tag{2}$$

where $\mathcal{C}(G)$ is the set of all cliques[1] in G and $Z(\theta)$ is the normalization constant (since it does not depend on x). Let C be a clique of G and $\mathcal{X}_C$ the corresponding joint domain of all vertices in C. The set Ω is the domain of the parameters $\theta \in \Omega^d$ which is usually the real line $\Omega = \mathbb{R}$. The parameter vector contains $|\mathcal{X}_C|$ weights for each clique $C \in \mathcal{C}(G)$, i.e. $\theta = (\theta_C)_{C \in \mathcal{C}(G)}$, which results in $d = \sum_{C \in \mathcal{C}(G)} |\mathcal{X}_C|$. The *compatibility functions* ψ_C (also known as *fac-*

[1] A clique corresponds to a fully connected subgraph.

tors) are typically chosen to be

$$\psi_C(x_C) = \exp(\langle \theta_C, \phi_C(x) \rangle)$$

since this ensures positivity of p_θ and leads to a canonical form of the corresponding exponential family member.

$$p_\theta(x) = \exp(\langle \theta, \phi(x) \rangle - A(\theta))$$

The vector-valued function ϕ is a *sufficient statistic* of X and may be understood as transformation of x into a binary valued feature space $\phi : \mathcal{X} \to \{0,1\}^d$ and $A(\theta) = \log Z(\theta)$. Sufficient statistics are called *overcomplete*, when there exists a vector $a \in \mathbb{R}^d$ and a constant $b \in \mathbb{R}$, such that $\langle a, \phi(x) \rangle = b, \forall x \in \mathcal{X}$. For convenience, the components of θ and ϕ are indexed by C to denote the subvector of weights or features that corresponds to a clique C. To address a certain component of θ or ϕ, the corresponding event $\{X_C = x_C\}$ is used as an index, i.e. $\theta_{X_C = x_C}$ or even $\theta_{C = x_C}$.

If the parameters are known, the MAP prediction for the joint state of all vertices can be computed by

$$x^* = \arg\max_{x \in \mathcal{X}} p_\theta(x) = \arg\max_{x \in \mathcal{X}} \langle \theta, \phi(x) \rangle . \quad (3)$$

A common choice for learning the parameters θ of an undirected model is the maximum likelihood estimation (MLE), where the likelihood (4) of the parameters θ for given i.i.d. data[2] $\mathcal{D}$ is maximized.

$$\mathcal{L}(\theta \mid \mathcal{D}) = \prod_{x \in \mathcal{D}} p_\theta(x) \quad (4)$$

The MLE θ^*, i.e. the solution that maximizes $\mathcal{L}$, has a closed form, if and only if the underlying graphical structure is a tree or a triangulated graph. In this case, θ^* is induced by the empirical expectation of the sufficient statistic $\phi(x)$.

$$\theta^*_{v=x} = \log \mathbb{E}_\mathcal{D}[\phi_{v=x}(x)], \quad (5)$$

$$\theta^*_{vu=xy} = \log \frac{\mathbb{E}_\mathcal{D}[\phi_{vu=xy}(x)]}{\mathbb{E}_\mathcal{D}[\phi_{v=x}(x)]\,\mathbb{E}_\mathcal{D}[\phi_{u=y}(x)]} \quad (6)$$

The MLE θ^* for partially observed data and certain classes of graphical models like Conditional Random Fields (CRF) (Sutton and McCallum, 2012) can be found with gradient based methods. Taking the logarithm of (4), dividing by $|\mathcal{D}|$ and substituting (1) for $p_\theta(x)$ yields the average log-likelihood ℓ (7). Since the logarithm is monotonic, maximizing ℓ will reveal the optimizer of $\mathcal{L}$. Since $\mathbb{E}_\mathcal{D}[\phi(x)] = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \phi(x)$, ℓ is given by

$$\ell(\theta \mid \mathcal{D}) = \langle \theta, \mathbb{E}_\mathcal{D}[\phi(x)] \rangle - \ln Z(\theta). \quad (7)$$

[2]It is assumed that every training instance in $\mathcal{D}$ is fully observed.

Taking the natural logarithm to form the log-likelihood is an arbitrary choice that may be replaced with any other $\log_b$ if desired. Since the second term is the cumulant generating function of p_θ, its partial derivative is the expected sufficient statistic for a given θ. This is plugged into the partial derivative of ℓ w.r.t. $\theta_{x_C = x_C}$ (7) to obtain the expression (8).

$$\frac{\partial \ell(\theta \mid \mathcal{D})}{\partial \theta_{x_C = x_C}} = \mathbb{E}_\mathcal{D}[\phi_{x_C = x_C}(x)] - \mathbb{E}_\theta[\phi_{x_C = x_C}(x)] \quad (8)$$

Here, $\mathbb{E}_\mathcal{D}[\phi_{x_C = x_C}(x)]$ denotes the empirical expectation of $\phi_{x_C = x_C}(x)$, i.e. its average value in $\mathcal{D}$. By using (8), the model parameters θ can be estimated by any first-order optimization technique. In the following, it is explained shortly how $\mathbb{E}_\theta[\phi_{x_C = x_C}(x)]$ is computed with BP. From now on, assume that the underlying graphical structure is a tree. The maximum clique size is thus 2. The message update rule is

$$m_{v \to u}(x_u) = \sum_{x_v \in \mathcal{X}_v} \psi_{vu}(x_v, x_u) \psi_v(x_v) \prod_{w \in \mathcal{N}_v \setminus \{u\}} m_{w \to v}(x_v). \quad (9)$$

The messages $m_{v \to u}(x_u)$ are computed for all pairs of vertex $v \in V$ and neighbor $u \in \mathcal{N}_v$ until convergence. Converged messages are denoted by $m^*_{v \to u}(x_u)$. The product of all incoming messages of a vertex is given by $M_v(x) := \prod_{u \in \mathcal{N}_v} m_{u \to v}(x)$. After convergence, the vertex marginal probabilities $p_v(x_v)$ that are implied by θ can be computed with

$$p_v(x_v) = \frac{\psi_v(x_v) M^*_v(x_v)}{\sum_{x \in \mathcal{X}_v} \psi_v(x_v) M^*_v(x)} \quad (10)$$

whereas $M^*_v(x)$ is the product of converged messages $m^*_{v \to u}(x_u)$. The interested reader is referred to (Kschischang et al., 2001) for a discussion on BP and related algorithms.

3 THE INTEGER APPROXIMATION

In their book on graphical models, Wainwright and Jordan (Wainwright and Jordan, 2007) stated that "It is important to understand that for a general undirected graph the compatibility functions ψ_C need not have any obvious or direct relation to marginal or conditional distributions defined over the graph cliques. This property should be contrasted with the directed factorization, where the factors correspond to conditional probabilities over the child-parent sets." This explains why it could be possible to have an undirected graphical model that is parametrized by integers. However, there is some work to do. For excluding every floating point computation, the identification of integer parameters is not enough. That is,

the computations for training and prediction have to be based on integer arithmetic. Lastly, integer approximation should still deliver a reasonable quality in terms of training error and test error.

The first step towards integer models is directly related to the above statement. Strictly speaking, the parameter domain Ω is restricted to the set of integers $\mathbb{N}$ and a new potential function is defined as

$$\overline{\psi}_C(x_C) := 2^{\langle \theta_C, \phi_C(x) \rangle} = \exp\left(\ln(2) \langle \theta_C, \phi_C(x) \rangle\right). \quad (11)$$

Considering parameters $\theta \in \mathbb{R}^d$ of a model that has potential function $\psi_C(x_C)$, it is easy to see that replacing $\psi_C(x_C)$ with $\overline{\psi}_C(x_C)$ does not alter the marginal probabilities as long as the parameters are scaled by $1/\ln 2$. By this, it is possible to convert integer parameters that are estimated with $\overline{\psi}_C(x_C)$ to $\psi_C(x_C)$ (and vice versa), without altering the resulting probabilities. Notice that $\overline{\psi}_C(x_C)$ can be computed by logical bit shift operations which consume which does consume less clock cycles than the corresponding transcendental functions required to compute $\psi_C(x_C)$.

As already mentioned above, it requires $\theta \in \mathbb{N}^d$ for $\overline{\psi}_C(x_C)$ to be integer and hence the product of compatibility functions in Eq. (1) and the normalization constant Eq. (2) are computable by means of non-negative integer arithmetic. By this, the probabilities that are computed by the model are rational, i.e. $p(x) \in [0,1] \cap \mathbb{Q}$ and may be represented as a pair of natural numbers. Nevertheless, actual probability values are not required at all for estimating the integer model parameters and to compute MAP predictions.

3.1 Inference

Recalling the message update Eq. (9), one sees that all messages are integer valued, if $\Omega \subseteq \mathbb{N}$, $\psi_C(x_C)$ is replaced by $\overline{\psi}_C(x_C)$ and the initial messages are set to 1. Thus, the whole message computation and propagation procedure is already stated without any floating point computation. Nevertheless, recall that the integer width of a CPU is constrained by its wordsize ω. $m_{v \to u}(x)$ may exceed the machine integer precision limit 2^ω quite easily. Thus, many overflows could occur during message computations which destroy the semantics of the messages and the results are no longer meaningful. First attempts to make the computation more robust against overflows relied on the fact that messages $m_{v \to u}(x)$ may be scaled arbitrarily without changing the resulting marginal probabilities as long as the same scale is used for all x. Nevertheless, the messages cannot be simply divided by their sum as is the case with floating point arithmetic, since integer division will pin all messages down to 0. Numerous attempts to scale the integer messages by bit shift operations only worked on relatively small graphical structures, but all those approaches suffered from the loss of information that occurred whenever too many bits had to be shifted out in order to prevent overflows.

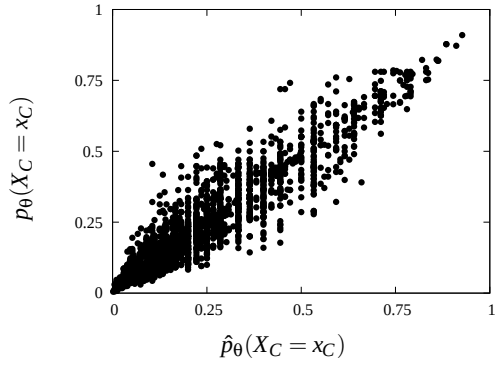

Figure 1: Estimates of edge marginal probabilities for 50 random trees with 50 nodes and 2 states per node. Marginals are computed by the bit length approximation ($\hat{p}$) and vanilla BP (p) on the same parameter vector θ.

As a solution to this problem, new messages are defined. Instead of computing the original sum-product messages, we propose to compute an approximation to the integer message bit length. The approximate bit length $\beta_{vu}(y)$ and the corresponding message $\hat{m}_{vu}(y) := 2^{\beta_{vu}(y)}$ are given by

$$\beta_{vu}(y) := \max_x \theta_{vu=xy} + \theta_{v=x} + \sum_{w \in \mathcal{N} \setminus \{u\}} \beta_{wv}(x). \quad (12)$$

How m and $\hat{m}$ are related to each other is a natural question. The messages $\hat{m}$ that result from the bit length approximation resemble max-product messages (Kschischang et al., 2001). Their magnitude is related to the original messages m through the following lemma.

Lemma 1. *Let $(v,u) \in E$ be an edge of tree $G = (V,E)$, $h_v := |\mathcal{X}_v|$ the size of vs state space and $n_v := |\mathcal{N}_v|$ the number of its neighbors. If $\forall y \in \mathcal{X}_u : \exists x \in \mathcal{X}_v : \theta_{vu=xy} + \theta_{v=x} > 0$, then $\hat{m}_{vu}(x) < m_{vu}(x) \leq \hat{m}_{vu}(x)^{h_v}$.*

This statement can be proven by induction over the degree n_v of v. The fact that the exclusion of all-0 vertex and edge parameters is a rather safe assumption, will be revealed later, when the integer parameter estimation is described.

When it comes to the point-wise estimates of the marginal probabilities, one finds that due to the approximate messages some marginal probabilities simply cannot be present. In Figure 1, edge marginal probabilities Eq. (10) are plotted that are computed with m and $\hat{m}$, respectively, while using the same parameters for both models. One clearly sees how the probability space is discretized by the approximate messages. One can also see that there is an error in the

approximate marginal probabilities computed with $\hat{m}$. In case of zero error all points would lie on the diagonal.

The previous lemma helps to derive an estimate of the distance between the true outcome of the inference and the one that results from the message update Eq. (12).

Theorem 1. *Let $\beta_v^* := \max_y \max_u \beta_{uv}(y)$ be the maximum incoming bit length at v and assume that the preconditions of Lemma 1 hold, then*

$$D(p_v \| \hat{p}_v) \in o\left(n_v h_v^2 \beta_v^*\right).$$

where $D(p_v\|\hat{p}_v)$ denotes the Kullback-Leibler (KL) divergence between the marginal probability mass function p_v, computed with the message update $m_{vu}(y)$, and $\hat{p}_v$, computed with $\hat{m}_{vu}(y)$.

This result can be derived by plugging the BP marginals (10) into the definition of the KL divergence and applying Lemma 1 two times. The KL is still unbounded, since there is no bound on β_v^*. Nevertheless, it indicates a dependence of the KL of p_v and $\hat{p}_v$ on the state space size $|\mathcal{X}_v|$ and the neighborhood size $|\mathcal{N}_v|$. This relation can also be observed in the numerical experiments in Section 4. A comprehensive discussion of how message errors affect the result of belief propagation can be found in (Ihler et al., 2005).

3.2 Parameter Estimation

In the following, an integer parameter estimation method based on the closed form solution to the MLE is derived. Recall that $\mathbb{E}_{\mathcal{D}}[\phi(x)] = \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}}\phi(x)$ and let $f := \sum_{x\in\mathcal{D}}\phi(x)$ and $\mathrm{bl}\, a := \lfloor \log_2 a \rfloor + 1$ the bit length of a. By approximating the logarithm in Eqs. (5) and (6) with bl, an integer approximation to the optimal parameters can be found:

$$\begin{aligned}\tilde{\theta}_{v=x} &:= \mathrm{bl}\, f_{v=x} - \mathrm{bl}\,|\mathcal{D}| \approx \log_2 \mathbb{E}_{\mathcal{D}}[\phi_{v=x}(x)] \\ \tilde{\theta}_{vu=xy} &:= \mathrm{bl}\, f_{vu=xy} - \mathrm{bl}\, f_{v=x} - \mathrm{bl}\, f_{u=y} + \mathrm{bl}\,|\mathcal{D}| \\ &\approx \log_2 \mathbb{E}_{\mathcal{D}}[\phi_{vu=xy}(x)]\end{aligned}$$

Unfortunately, most of those estimates are negative which is not allowed due to the integer restriction. In case of negative parameters, let $\lambda := |\max_{1\le i\le d} -\tilde{\theta}_i|$ be the absolute value of the smallest component of $\tilde{\theta}$. Now, consider the weights

$$\tilde{\theta}^+_{v=x} := s + \tilde{\theta}_{v=x}, \qquad \tilde{\theta}^+_{vu=xy} := s + \tilde{\theta}_{vu=xy}$$

with $s := (\lambda, \lambda, \ldots, \lambda)^\top \in \mathbb{R}^d$. The new parameters $\tilde{\theta}^+$ are non-negative, but an error is induced into $\tilde{\theta}$ by replacing $\log_2$ with bl. The following lemma shows that shifting $\tilde{\theta}$ by s introduces no new error.

Lemma 2. *Let $s := (c, c, \ldots, c)^\top \in \mathbb{R}^d$ with arbitrary c and let ϕ be an overcomplete sufficient statistic, then $\ell(\theta + s) = \ell(\theta)$.*

The statement follows from the definitions of log-likelihood and overcompleteness. This shows that it is safe to assume $\forall y \in \mathcal{X}_u : \exists x \in \mathcal{X}_v : \theta_{vu=xy} + \theta_{v=x} > 0$ for Lemma 1, since we can enforce all parameters to be positive without touching the likelihood. It is also possible to bound the training error of the shifted integer parameters $\tilde{\theta}^+$:

Theorem 2. *Let θ^* be as defined by Eqs. (5) and (6) and $\tilde{\theta}^+$ as defined above, then $\ell(\theta^*) - \ell(\tilde{\theta}^+) \le \|\nabla f(\tilde{\theta}^+)\|_1$.*

The result follows from the previous lemma and the definition of convexity.

Either due to restrictions in wordsize ω or for enlarging the number of representable marginal probabilities, a final scaling of the parameters might be desired. To allow an appropriate integer scaling, a parameter $K := |\Omega|$ is introduced: Let $\gamma := \max_{1\le i\le d} -\tilde{\theta}_i$ be the negative component of $\tilde{\theta}$ with the largest magnitude and $\kappa := \max_{1\le i\le d} \tilde{\theta}_i$ be the component of $\tilde{\theta}$ with the largest magnitude. The final integer parameters are computed by

$$\bar{\theta}_{v=x} := \left\lfloor \frac{K}{\kappa - \gamma} \tilde{\theta}^+_{v=x} \right\rfloor, \bar{\theta}_{vu=xy} := \left\lfloor \frac{K}{\kappa - \gamma} \tilde{\theta}^+_{vu=xy} \right\rfloor. \tag{13}$$

Thus, $\tilde{\theta}^+$ is rescaled such that $\bar{\theta} \in \{0, 1, \ldots, K\}^d$, which may also be interpreted as implicit base change. Note that unless $K = (\kappa - \gamma)$, the parameter vector is scaled and an additional approximation error is added. Hence, the impact of K is empirically evaluated in Section 4. The method of choosing parameters according to (13) is called *direct integer estimation*.

3.3 Gradient based Estimation

As mentioned in Section 2, in certain situations, it might be desired to estimate the parameters with gradient based methods. Unfortunately, the partial derivatives from Eq. (8) are not integral. The expression must be rearranged to obtain an integer form. Let $f := \sum_{x\in\mathcal{D}}\phi(x)$ and $\hat{\mathcal{M}}_v^* := \sum_{y\in\mathcal{X}_v}\hat{M}_v^*(y)$, it is

$$\hat{\mathcal{M}}_v^* |\mathcal{D}| \frac{\partial \ell(\theta \mid \mathcal{D})}{\partial \theta_{X_v = x_v}} = \hat{\mathcal{M}}_v^* f_{v=x_v} - |\mathcal{D}|\, \hat{M}_v^*(x_v).$$

This scaled version of the partial derivative is an integer expression that can be computed by using only integer addition, multiplication and binary bit shift. The common gradient descent update makes use of a stepsize η to determine how far the current weight

vector should move into the direction of the gradient. The smallest possible step size in integer space is 1. This means that any parameter can either be increased or decreased by 1. In the beginning of an integer gradient based optimization, the gradient will tell to increase a quite large number of parameters. This results in rather slow convergence, since due to the fixed step size of 1, most of the parameters are worse than before the update. To compensate for this, we suggest to update, for each clique, only the parameter for which the corresponding partial derivative has the largest magnitude. This method is used when estimating the CRF parameters in the following section.

4 NUMERICAL RESULTS

The previous sections pointed out various factors that may have an influence on training error, test performance or runtime of the integer approximation. In order to show that integer undirected models are a quite general approach for approximate learning in discrete state spaces, generative and discriminative variants of undirected models are evaluated on synthetic data and real world data. In particular the following methods are considered: RealMRF: The classic generative undirected model as described in Section 2. RealCRF: The discriminative classifier as it is defined in (Lafferty et al., 2001; Sutton and McCallum, 2012). IntMRF: The integer approximation of generative undirected models as described in Section 3. IntCRF: The integer approximation of discriminative undirected models. Further details are explained in Section 4.4. Both real variants are based on floating point arithmetic. In the MRF experiments, the model parameters are estimated from the empirical expectations by Eqs. (5), (6) and (13). Parameters of discriminative models are estimated by stochastic gradient methods (Sutton and McCallum, 2012). Each MRF experiment was repeated 100 times on random input distributions and graphs. In most cases, only the average is reported, since the standard deviation was too small to be visualized in a plot. Whenever MAP accuracy is reported, it corresponds to the percentage of correctly labeled vertices, where the prediction is computed with Eq. (3).

The implementations[3] of all evaluated methods are equally efficient, e.g. the message computation (and therefore the probability computation) executes exactly the same code, except for the arithmetic instructions. Unless otherwise explicitly stated, the experiments are done on an Intel Core i7-2600K 3.4GHz (Sandy Bridge architecture) with 16GB 1333MHz DDR3 main memory.

Synthetic Data. In order to achieve robust results that capture the *average behavior* of the integer approximation, a synthetic data generator has been implemented that samples random empirical marginals with corresponding MAP states. Therefore, a sequential algorithm for random trees with given degrees (Blitzstein and Diaconis, 2011) generates random tree structured graphs. For a random graph, the weights $\theta_i^* \sim \mathcal{N}(0,1)$ are sampled from a Gaussian distribution. Additionally, for each vertex, a random state is selected that gets a constant extra amount of weight, thus enforcing low entropy. The weights are then used to generate marginals and MAP states with the double precision floating point variant of belief propagation. The generated marginals serve as empirical input distribution and the MAP state is compared to the MAP state that is estimated by IntMRF and RealMRF.

CoNLL-2000 Data. This data set was proposed for the shared task at the Conference on Computational Natural Language Learning in 2000 and is based on the Wall Street Journal corpus. The latter contains word-features and one label, called chunk tag, per word. In total, there are 22 chunk tags that correspond to the vertex states, i.e. $|\mathcal{X}| = 22$. For the computation of per chunk F_1-score, a chunk is only treated as correct, if and only if all consecutive tags that belong to the same chunk are correct. The data set contains 8936 training instances and 2012 test instances. Because of the inherent dependency between neighboring vertex states, this data set is well suited to evaluate whether the dependency structure between vertices is preserved by the integer approximation.

4.1 The Impact of $|\mathcal{X}|$ and $|\mathcal{N}_v|$ on Quality and Runtime

In Section 3 an estimate of the error in marginal probabilities that are computed with bit length BP (Section 3.1) indicates that the size of the vertex state space $|\mathcal{X}_v|$ and the degree $|\mathcal{N}_v|$ have an impact on the training error. In Figure 2, the training error in terms of normalized negative log-likelihood, the test error in terms of MAP accuracy and the runtime in seconds for two values of $|\mathcal{X}_v|$ and $|\mathcal{N}_v|$ for an increasing number of vertices on the synthetic data are shown. Each point in each curve is the average over 100 random trees with random parameters. The results with varying $|\mathcal{X}_v|$ are generated with a maximum degree of 8 and the ones for varying $|\mathcal{N}_v|$ with $|\mathcal{X}_v| = 4$.

In terms of training error, the mid-right plot shows a clear offset between integer and floating point estimates for the same number of states. In terms of

[3]For reproducibility, all data and code is available at http://sfb876.tu-dortmund.de/intmodels.

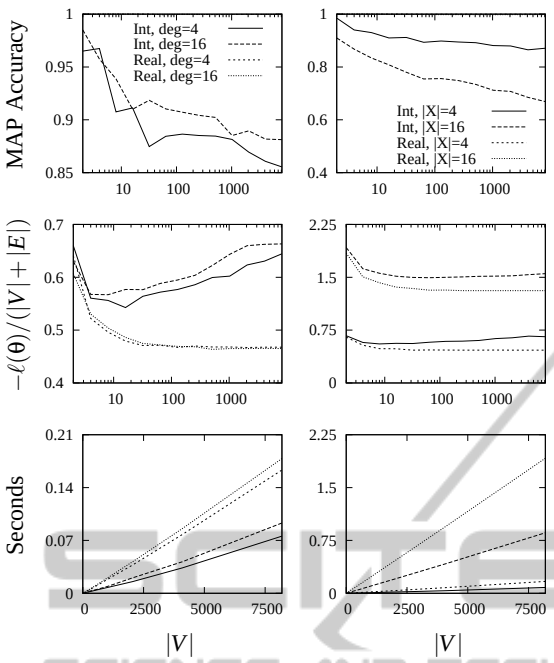

Figure 2: MRF test accuracy (top), training error (center) and runtime in seconds (bottom) for different choices of maximum vertex degree (left column) and state space size (right column) as a function of the number of vertices (x-axis). The plots in each column share the same key that is defined in the top row. The plots in the first two rows have their x-axis in logarithmic scale. IntMRF results are generated with $K = 8$.

varying degrees (mid-left), the training error of the integer model responds to different neighborhood sizes whereas the likelihood of the floating point model is invariant against the degrees. A similar picture is drawn for the dependence of the test accuracy with $|\mathcal{X}|$ in the top-left and $|\mathcal{N}_v|$ in the top-right plot, respectively. The floating point MAP estimate is always correct and not changed by an increasing number of states and neighbors, whereas the performance of IntMRF drops with an increasing number of vertices but actually increases with increasing degrees.

The curves that correspond to RealMRF are not visible in the top row of Figure 2, since this model gets 100% accuracy on the synthetic data in nearly all runs and therefore, its curve lies close to the horizontal 1-line. However, in the two plots at the bottom of Figure 2, one can see that the resource consumption in terms of clock cycles is largely reduced by the integer model. The time therein is measured for estimating parameters, computing the likelihood, and performing a MAP prediction. Since both algorithms (RealMRF and IntMRF) share exactly the same asymptotic complexity for these procedures, the substantial reduction in runtime that is shown in the results must be due to the reduction in clock cycles.

4.2 The Contribution of K to Quality, Gradient and Memory Consumption

As described in Section 3.2, the integer parameter vector $\bar{\theta}$ is scaled to be in the set $\Omega_K^d := \{0,1,\ldots,K\}^d$. The effect of such scaling is illustrated by the response of the integer model to the magnitude of K in terms of training quality and test error, shown in the two plots on the top of Figure 3. The training error seems to be a quite smooth function of K whereas the MAP accuracy is sensitive to choices of K. This could be expected, since a large K means that a larger number of marginal probabilities can be represented. One can also see that, as soon as K is large enough, (i.e. $K = 8$ in the first two plots in Figure 3) a further increase in K does not show any significant impact on either training error and test accuracy. Both results are generated on graphs with a maximum degree of 8, but as already known from the previous experiment, the effect of different degrees on the model quality is negligible. In the third plot of Figure 3, the width of the intrinsic parameter space is shown, i.e. the difference of the largest and the smallest parameter before rescaling. The difference $(\kappa - \gamma)$ seems to converge to the same value for various configurations of n and $\mathcal{X}$. Plotting κ and γ separately shows that the dynamics in $(\kappa - \gamma)$ are mainly influenced by the smallest parameter γ, i.e., the width of the parameter space must increase in order to represent small probabilities.

As indicated by the analysis of the training error in Section 3, the distance between the maximum likelihood estimate and the result of the direct integer parameter estimation is bounded by the gradient norm of the integer parameters $\bar{\theta}$. Since the components of the gradient are differences of probabilities, their value cannot exceed 1 and a trivial upper bound for the ℓ_1-norm of the gradient is therefore d. An important observation can be made in the rightmost picture of Figure 3 which shows the relative gradient norm for an increasing number of vertices and various values K. This result suggests that there exists a bound on the relative gradient norm that is independent of the number of vertices and that this bound decreases with increasing K.

Furthermore, K has a strong impact on the overall memory consumption of the model. Let ω be the floating point wordsize of the underlying architecture. The Real model will require $d \times \omega$ bit, whereas the Int model has a size of $d \times \mathrm{bl}K$ (cf. Section 3.2) bit. Thus, the Int model will use less memory whenever $\mathrm{bl}K < \omega$. In practice, a specialized data structure is required to observe the reduced memory consumption. In the second plot of Fig. 3 and the last two plots

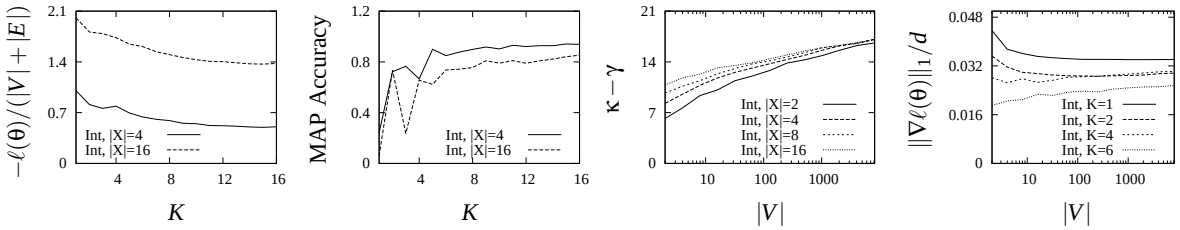

Figure 3: From left to right: (I) negative log-likelihood of IntMRF as a function of K, i.e. the size of the parameter domain. (II) MAP accuracy of MRF as a function of K. (III) Natural size of the parameter domain as a function of the number of vertices (in log-scale) for different state space sizes, whereas γ is the smallest and κ the largest element of the corresponding estimated parameter vector. (IV) The relative ℓ_1-norm of the gradient for various values of K as a function of the number of vertices (in log-scale).

of Fig. 4, one can see that $K \in \{6,8\}$ seems to be a reasonable choice for these data sets, which reduces the memory consumption of the model by a factor of 20, compared to a 64 bit floating point model.

4.3 Integer Models on Resource Constrained Devices

The motivation for the integer model was to save resources in terms of clock cycles. Here, it is demonstrated that the impact of this reduction is larger, if the underlying architecture is weaker, i.e. has slower floating point arithmetic. A runtime comparison of the integer MRF on two different CPU architectures is shown in Figure 4. One is the Sandy Bridge, which has also been the platform for all the other experiments, and the second one is a Raspberry Pi device with the ARM11 architecture. As expected, the integer model actually speeds up the execution on the Pi device more than on the other architecture, i.e. the Raspberry Pi gains a speedup of $2.56\times$ and the Sandy Bridge a speedup of $2.34\times$. In terms of standard deviation, the ARM11 architecture is more stable than the Sandy Bridge, which might be a result of a more sophisticated out-of-order execution in the latter architecture.

4.4 Training Integer CRF with Stochastic Integer Gradient Descent

In the last evaluation, discriminative models for chain structured data are investigated. A linear-chain CRF is constructed on the CoNLL-2000 data and trained by a SGD algorithm. The floating point CRF is trained with stepsize $\eta = 10^{-1}$. Integer parameter updates are computed by means of the scaled integer gradient (cf. the end of Section 3). Both algorithms perform 20 passes over the training data, each pass looping through the training instances in random order, whereby IntCRF was trained with three different parameter domain sizes $K \in \{2,4,6\}$. This was repeated 50 times in order to compute an estimate of the expected quality of the randomized training procedure. Training error and test error are reported in Figure 4 as a function of runtime in seconds. The negative log-likelihood is averaged over all training instances and the accuracy is computed w.r.t. the chunk type. One clearly sees that $K = 2$ results in poor training and test performances. Nevertheless, IntCRF with $K \in \{4,6\}$ reaches nearly the same performance as the double precision model, while using less computational resources per SGD iteration. Furthermore, the average precision, recall and F_1-score were computed for each chunk type. As desired, the performance of the integer approximation is reasonable. Except for one chunk type (VP), the difference in F_1-score between both methods is below 5%. Surprisingly, the IntCRF has a higher precision than the RealCRF for three chunk types. Averaged over all 22 chunk types, the IntCRF has $< 1\%$ less precision, recall and F_1-score than RealCRF.

5 CONCLUSIONS AND FUTURE WORK

In this work, integer undirected graphical models have been introduced together with algorithms for probabilistic inference and parameter estimation that rely only on integer arithmetic. Generative and discriminative models have been evaluated in terms of prediction quality and runtime. On different architectures, an average speedup of at least $2\times$ can be achieved while accepting a reasonable loss in accuracy. One of the findings from the empirical evaluation was the fact, that the model parameters have a rather small magnitude. Thus, only a few bits are required for each parameter, which reduces the models size by more than $20\times$ compared to the eight bytes that are required to store 64 bit double precision parameters. Overall, the results show that our method is

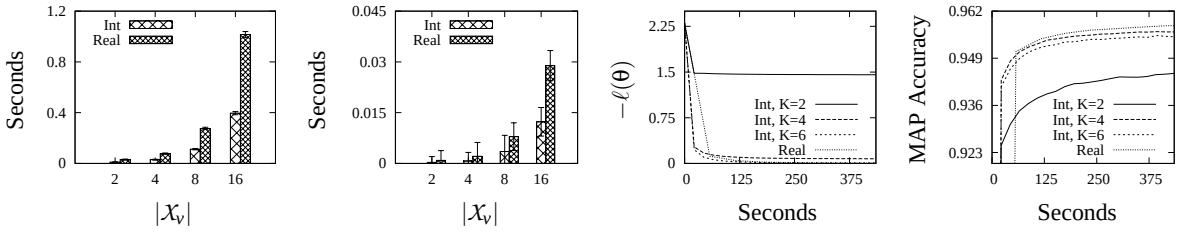

Figure 4: From left to right: Runtime comparison of integer and floating point MRF on two architectures for a varying number of states. (I) Raspberry PI @ 700MHz (ARM11). (II) Intel Core i7-2600K @ 3.4GHz (Sandy Bridge). Progress of stochastic gradient training in terms of (III) training error and (IV) test accuracy as a function of runtime.

especially well suited for small, mobile devices. Beside the introduction of integer models, average runtimes of probabilistic models have been investigated and optimized in the context of resource constraint devices. The experimental results indicate that a tight bound on the training error might exist. Therefore, future work will focus on tighter error bounds.

ACKNOWLEDGEMENTS

This work has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 “Providing Information by Resource-Constrained Data Analysis”, projects A1 and C1.

REFERENCES

Ahmadi, B., Kersting, K., and Natarajan, S. (2012). Lifted online training of relational models with stochastic gradient methods. In *Machine Learning and Knowledge Discovery in Databases*, volume 7523 of *Lecture Notes in Computer Science*, pages 585–600.

Blitzstein, J. and Diaconis, P. (2011). A sequential importance sampling algorithm for generating random graphs with prescribed degrees. *Internet Mathmatics*, 6(4):489–522.

Choirat, C. and Seri, R. (2012). Estimation in discrete parameter models. *Statistical Science*, 27(2):278–293.

Hassibi, A. and Boyd, S. (1998). Integer parameter estimation in linear models with applications to gps. *IEEE Transactions on Signal Processing*, 46(11):2938–2952.

Ihler, A. and McAllester, D. (2009). Particle belief propagation. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, pages 256–263, JMLR: W&CP 5.

Ihler, A. T., III, J. W. F., and Willsky, A. S. (2005). Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6:905–936.

Kersting, K., Ahmadi, B., and Natarajan, S. (2009). Counting belief propagation. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 277–284,

Kschischang, F. R., Frey, B. J., and Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289

Noorshams, N. and Wainwright, M. (2011). Stochastic belief propagation: Low-complexity message-passing with guarantees. In *Communication, Control, and Computing 49th Annual Allerton Conference on*, pages 269–276.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Sontag, D. and Jaakkola, T. (2009). Tree block coordinate descent for MAP in graphical models. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics* , volume 8, pages 544–551. JMLR: W&CP.

Sutton, C. and McCallum, A. (2012). An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4):267–373.

Tschiatschek, S., Reinprecht, P., Mücke, M., and Pernkopf, F. (2012). Bayesian network classifiers with reduced precision parameters. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*.

Wainwright, M. J., Jaakkola, T. S., and Willsky, A. S. (2003). Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching. In *9th Workshop on Artificial Intelligence and Statistics*.

Wainwright, M. J. and Jordan, M. I. (2007). Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305.