Scaled Conjugate Gradients for Maximum Likelihood: An Empirical Comparison with the EM Algorithm

Kristian Kersting **Niels Landwehr**
Institute for Computer Science, Machine Learning Lab
Albert-Ludwigs-University, Georges-Köhler-Allee, Gebäude 079,
D-79085 Freiburg i. Brg., Germany
{kersting,landwehr}@informatik.uni-freiburg.de

Abstract

To learn Bayesian networks, one must estimate the parameters of the network from the data. EM (Expectation-Maximization) and gradient-based algorithms are the two best known techniques to estimate these parameters. Although the theoretical properties of these two frameworks are well-studied, it remains an open question as to when and whether EM is to be preferred over gradients. In this paper, we answer this question empirically. More specifically, we first adapt scaled conjugate gradients well-known from neural network learning. This accelerated conjugate gradient avoids the time consuming line search of more traditional methods. Secondly, we empirically compare scaled conjugate gradients with EM. The experiments show that accelerated conjugate gradients are competitive with EM. Although, in general EM is the domain independent method of choice, gradient-based methods can be superior.

1 Introduction

Bayesian networks [1] are one of the most important, efficient and elegant frameworks for representing and reasoning with probabilistic models. They specify joint probability distributions over finite sets of random variables, and have been applied to many real-world problems in diagnosis, forecasting, automated vision, sensor fusion and manufacturing control. Over the past years, there has been much interest in the problem of learning Bayesian networks from data. This problem is crucial in many applications in a wide range of domains. It can be specified as follows. Given a finite set of independently drawn data cases, construct a Bayesian network that best models the data.

The problem of *parameter estimation* is a fundamental task not only because of the difficulty the inability of humans to reliably estimate the parameters, but also because of it forms the basis for the overall learning problem [2]. A classical, frequentist parameter estimation method is *maximum likelihood estimation*. Here, one selects those parameters maximizing the likelihood (i.e. the probability of the observed data as a function of the unknown parameters) computed according to the current model. Unfortunately in many real-world domains, the data cases available are incomplete, i.e. some values may not be observed. In presence of missing data, the maximum likelihood estimate typically cannot be written in closed form. It is a numerical optimization problem, and all known algorithms involve nonlinear optimization and multiple calls to a Bayesian network inference as subroutines. The most prominent techniques within Bayesian networks are the EM (Expectation and Maximization) algorithm [3, 4] and gradient-based approaches [5, 6, 7].

The comparison between EM and (advanced) gradient techniques like conjugate gradient is not well understood. Both methods perform a greedy local search which is guaranteed to converge to stationary points. They both exploit expected sufficient statistics as their primary computation step. However, there are important differences. The EM is easier to implement, converges much faster than simple gradient, and is somewhat less sensitive to starting points. (Conjugate) gradients estimate the step size (see below) with a line search involving several additional Bayesian network inferences compared to EM. But, gradients are more flexible than EM, as they allow e.g. to learn non-multinomial parameterizations using the chain rule for derivatives [5] or to choose other scoring functions than the likelihood [6]. The EM algorithm may slowly converge, but can be speed up near the maximum using gradient-based approaches [8, 9]. Other acceleration approaches execute only a partial maximization step of the EM algorithm based on gradient techniques leading to generalized EM algorithms [8, 10, 11, 9].

In this paper, we report on an experimental comparison between EM and (accelerated) conjugate gradients to maximum likelihood parameter estimation. In doing so, we follow Ortiz and Kaelbling's [9] suggestion for such a comparison in the context of (discrete) Bayesian network. To overcome the expensive line search, we adapted *scaled conjugate gradients* (SCG) [12] from the field of learning neural networks. They avoid the line search by using a Levenberg-Marquardt approach [13] in order to scale the step size. This type of accelerated conjugate gradients are novel in the context of Bayesian networks. Other work investigated approximated line searches only to accelerate EM [8, 9, 10, 11]. From the experimental results, we will argue that scaled conjugate gradients are superior to traditional conjugate gradients. They incorporate advantages of the EM algorithm into gradient-based algorithms making them competitive with EM. The EM algorithm seems to be the best domain independent choice, though (accelerate) gradient methods can be superior depending on properties of the domain such as the fraction of latent parameters and the prior information encoded the network structure.

We proceed as follows. After briefly introducing Bayesian networks in Section 2, Section 3 reviews the EM algorithm and the basic (conjugate) gradient formulae. It then shows how to adapt scaled conjugate gradients. We experimentally compare these algorithms in Section 4. We discuss related work in Section 5.

2 Bayesian networks

Throughout the paper, we will use X to denote a random variable, x a state and $\mathbf{X}$ (resp. $\mathbf{x}$) a vector of variables (resp. states). A *Bayesian network* [1] represents the joint probability distribution $\mathbf{P}(X_1, \ldots, X_n)$ over a set $\mathbf{X} = \{X_1, \ldots, X_n\}$ of random variables. In this paper, we restrict the X_i's to be finite random variables, i.e. each X_i has a finite set $x_i^1, \ldots, x_i^{k_i}$ of possible states. A Bayesian network is an augmented, acyclic graph, where each node corresponds to a random variable X_i and each edge indicates a direct influence among the random variables. We denote the parents of X_i in a graph-theoretical sense by $\mathbf{Pa}_i$. The family of X_i is $\mathbf{Fa}_i := \{X_i\} \cup \mathbf{Pa}_i$ To each node X_i, a conditional probability table[1] (CPT) is associated specifying the distribution $\mathbf{P}(X_i \mid \mathbf{Pa}_i)$. The table entries are $\theta_{ijk} = P(X_i = x_i^k \mid \mathbf{Pa}_i = \mathbf{pa}_i^j)$, The network stipulates the assumption that each node X_i in the graph is conditionally independent of any subset $\mathbf{A}$ of nodes that are not descendants of X_i given a joint state of its parents. Thus, the joint probability distribution over $\mathbf{X}$ factors to $\mathbf{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbf{P}(X_i \mid \mathbf{Pa}_i)$. In the rest of the paper, we will represent a Bayesian network by the vector θ consisting of all θ_{ijk}'s.

[1] There are more sophisticated representation, but we will not discuss them here.

3 Maximum Likelihood Estimation

Here, we will show how to estimate the maximum likelihood parameters of a given Bayesian network. Let $\mathbf{D} = \{\mathbf{d_1}, \ldots, \mathbf{d_N}\}$ be a set of data cases, i.e. partial joint states over $\mathbf{X}$. We assume that the data cases are independently sampled from identical distributions (idd). The likelihood $L(\mathbf{D}, \theta)$ is the probability of the observed data $\mathbf{D}$ as a function of the unknown parameters θ, i.e. $L(\mathbf{D}, \theta) := P(\mathbf{D} \mid \theta)$ which simplifies because of the idd assumption to $L(\mathbf{D}, \theta) = \prod_{l=1}^{N} P(\mathbf{d}_l \mid \theta)$. Thus, the search space $\mathcal{H}$ is spanned by the space over the possible values of θ and we seek to find $\theta^* = \max_{\theta \in \mathcal{H}} L(\mathbf{D}, \theta)$. Due to the monotonicity of the logarithm, this yields

$$\theta^* = \max_{\theta \in \mathcal{H}} \sum_{l=1}^{N} \log P(\mathbf{d}_l \mid \theta).$$

An important issue is, whether the data $\mathbf{D}$ is complete, or not. In the case of complete data, i.e. the values of all random variables are observed, Lauritzen [4] showed that maximum likelihood estimation simply corresponds to frequency counting in the following way. Let $n(\mathbf{a} \mid \mathbf{D})$ denote the *counts* for a particular joint state $\mathbf{a}$ of variables $\mathbf{A}$ in the data cases, then

$$\theta_{ijk}^* = \frac{n(\mathbf{fa}_i^k \mid \mathbf{D})}{n(\mathbf{pa}_i^j \mid \mathbf{D})}. \tag{1}$$

However, in the presence of missing data, the maximum likelihood estimate typically cannot be written in closed form, and iterative optimization schemes like the EM or conjugate gradient algorithms are needed.

3.1 EM Algorithm

The Expectation-Maximization algorithm [3, 4] is a standard tool for maximum likelihood estimation in the presence of incomplete data. Its basic idea is that if we know the values for all nodes, learning would be easy. It iteratively performs two steps: First, based on the current parameters θ^n and the observed data the expected values are computed using an inference algorithm. Then it treats the expected values as though they were observed, computing new parameters θ^{n+1}. Lauritzen [4] showed that this idea leads to a modified equation (1) where *expected counts* $\bar{n}(\mathbf{a}) = \sum_{l=1}^{N} P(\mathbf{a} \mid \mathbf{d}_l, \theta^n)$ are used instead of *counts*. To compute $P(\mathbf{a} \mid \mathbf{d}_l, \theta^n)$ an Bayesian network inference engine can be use.

3.2 Gradient-Ascent Learning

Gradient ascent, also known as *hill climbing*, is a classical method for finding a maximum of an (scoring) function. It iteratively performs two steps.

1. It computes the *gradient* vector ∇_θ of partial derivatives of the scoring function with respect to the parameters of a Bayesian network at a given point $\theta \in \mathcal{H}$.

2. Then it takes a small step in the direction of the gradient to the point $\theta + \delta \nabla_\theta$ where δ is the step-size parameter.

The algorithm will converge to a local maximum for small enough δ (cf. [13]). Thus, we have to compute the partial derivatives of $P(\mathbf{D} \mid \theta)$ w.r.t. parameters θ_{ijk}. According to Binder *et al.* [5], they are given by

$$\frac{\partial logP(\mathbf{D} \mid \theta)}{\partial \theta_{ijk}} = \sum_{l=1}^{N} \frac{P(\mathbf{fa}_i \mid \mathbf{d}_l, \theta)}{\theta_{ijk}} \tag{2}$$

Again, $P(\mathbf{fa}_i \mid \mathbf{d}_l, \theta)$ can be computed using an inference engine.

3.2.1 Constraint Satisfaction

In the problem at hand, the described method has to be modified to take into account the constraint that the parameter vector θ consists of probability values, i.e. $\theta_{ijk} \in [0,1]$ and $\sum_j \theta_{ijk} = 1$. There are two [5] ways to enforce this. (1) projecting the gradient onto the constraint surface, and (2) reparameterizing the problem. We choose the latter approach using $\theta_{ijk} = (\beta_{ijk}^2)/(\sum_l \beta_{ilk}^2)$ because the reparameterized problem is fully unconstrained. Applying the chain rule of derivatives yields

$$\frac{\partial logP(\mathbf{D}\,\theta)}{\partial \beta_{ijk}} = \sum_{i'j'k'} \frac{\partial logP(\mathbf{D} \mid \theta)}{\partial \theta_{i'j'k'}} \cdot \frac{\partial \theta_{i'j'k'}}{\partial \beta_{ijk}} \tag{3}$$

Since $\partial \theta_{i'j'k'} / \partial \beta_{ijk} = 0$ unless $i = i'$ and $j = j'$, equation (3) simplifies to

$$\begin{aligned}
\frac{\partial logP_w(\mathbf{D})}{\partial \beta_{ijk}} &= \sum_{j'} \frac{\partial logP(\mathbf{D} \mid \theta)}{\partial \theta_{ij'k}} \cdot \frac{\partial \theta_{ij'k}}{\partial \beta_{ijk}} \\
&= \sum_{j'} \frac{\partial logP(\mathbf{D} \mid \theta)}{\partial \theta_{ij'k}} \cdot \frac{\frac{\partial \beta_{ij'k}^2}{\partial \beta_{ijk}} \sum_l \beta_{ilk}^2 - \frac{\partial \sum_l \beta_{ilk}^2}{\partial \beta_{ijk}} \beta_{ij'k}^2}{(\sum_l \beta_{ilk}^2)^2} \\
&= \frac{2\beta ijk \frac{\partial logP(\mathbf{D}|\theta)}{\partial \theta_{ijk}} \sum_l \beta_{ilk}^2 - \sum_l \frac{\partial logP(\mathbf{D}|\theta)}{\partial \theta_{ilk}} \beta_{ilk}^2}{(\sum_l \beta_{ilk}^2)^2}.
\end{aligned}$$

3.2.2 Conjugate Gradients

As already noted, the simple gradient ascent algorithm uses a fixed step size δ when following the gradient. As a consequence, it often converges very slowly. Though it will converge to a (local) maximum for small enough δ, it is not a priori clear how to choose δ. Instead, it would be better to perform a series of *line searches* to choose δ, i.e. to do a one dimensional iterative search for δ in the direction of ∇_{θ_i} maximizing $L(\mathbf{D}, \theta + \delta \cdot \nabla_{\theta_i})$. The problem with the resulting gradient ascent algorithm is, that a maximization in one direction could spoil past maximizations. This problem is solved in *conjugate gradient* methods (see e.g [13]) by computing so-called conjugate directions $h_0, h_1, h_2, \ldots$ which are not interfering. As Nocedal [14] noted, the best variant is generally believed to be the *Polak-Ribiere* conjugate gradient. Following the schema of hill climbing, it iteratively performs two steps starting with $\theta_0 \in \mathcal{H}$ and $h_0 = \nabla_{\theta_0}$:

1. (Conjugate directions) Set the next direction $h_{i+1} = \nabla_{\theta_{i+1}} + \gamma_i \cdot h_i$ where
$$\gamma_i = \frac{(\nabla_{\theta_{i+1}} - \nabla_{\theta_i}) \cdot \nabla_{\theta_{i+1}}}{\nabla_{\theta_i} \cdot \nabla_{\theta_i}}.$$

2. (Line search) Compute θ_{i+1} by maximizing $L(\mathbf{D}, \theta_i + \delta \cdot h_{i+1})$ in the direction of h_{i+1}.

For a detailed discussion, we refer to e.g. [13, 14].

3.2.3 Scaled Conjugate Gradients

The estimation of the step size in conjugate gradients is done with a line search. A common technique is to initially bracketing a maximum and then isolating it using an exact algorithm like Brent's Method, but other methods are possible (see e.g. [15]). There are several drawbacks of doing a line search. First, it introduces new problem-dependent parameters such as stopping criterion. Second, the line search involves several likelihood evaluations, i.e. network inferences which are known to be NP-hard [16]. Thus, the line search dominates the computational costs of conjugate gradients resulting in a serious disadvantage compared to the EM. The latter one competes with only one inference per iteration.

Therefore, it is not surprising that researchers in the UAI community used inexact line search to reduce the complexity [8, 9, 11] when accelerating EM. However, the use of both exact and inexact line searches within conjugate gradients requires careful considerations [9, 14]. Thus, we decided to use a variant of conjugate gradients called *scaled conjugate gradients* due to Møller [12] which led to significant speed up

Set *success* = *true*;
while success = true *and* λ_j *is less some threshold* **do**
  Compute gradient and second order information;
  Scale β_{j+1} using (6);
  if $\beta_{j+1} \leq 0$ **then** make Hessian positive definite;
  Calculate new step size δ_{j+1};
  if *succesful reduction in error can be made* **then**
    Set *success* = *true*;
    Compute new conjugate direction h_{j+1} and $\mathbf{v}_{j+1}$;
  else set *success* = *false*;

Algorithm 1: A simplified scheme of scaled conjugate gradients.

in the context of learning neural networks while preserving accuracy. The line search is substituted by a scaling of the step size δ depending on success in error reduction and goodness of a quadratic approximation of the likelihood. The approximation costs one additional inference compared to EM.

Scaled conjugate gradients (see Algorithm 1) differ from the general scheme of conjugate gradients (see above) mainly in the *line search* step. Due to lack of space, we will only sketch the algorithm, details can be found in [12], and for the sake of simplicity, we will illustrate the idea minimizing a general function $f : \mathbb{R}^k \mapsto \mathbb{R}$. Substituting f by $-f$ yields the corresponding maximization. Let

$$\tilde{f}_{\mathbf{w}}(\mathbf{x}) = f(\mathbf{w}) + f'(\mathbf{w})^T \cdot \mathbf{x} + \frac{1}{2}\mathbf{x}^T \cdot f''(\mathbf{w}) \cdot \mathbf{x}$$

be the quadratic approximation of f at $\mathbf{w}$, $\mathbf{v}_j$ the last parameter estimation, and h_{j+1} the current conjugate direction. Then, the step size δ_{j+1} minimizing $\tilde{f}_{\mathbf{v}_j}(\mathbf{v}_j + \delta \cdot h_{j+1})$ is given by

$$\delta_{j+1} = \frac{-h_{j+1}^T \cdot \tilde{f}'_{\mathbf{v}_j}(\mathbf{v}_j)}{h_{j+1}^T \cdot f''(\mathbf{v}_j) \cdot h_{j+1}}, \tag{4}$$

provided that the Hessian $f''(\mathbf{v}_j)$ is positive definite. Assuming this, the next parameter estimation $\mathbf{v}_{j+1}$ would be $\tilde{f}_{\mathbf{v}_j}(\mathbf{v}_j + \delta_{j+1} \cdot h_{j+1})$. However, the computation of the second order term $f''(\mathbf{v}_j) \cdot h_{j+1}$ in (4) is expensive. It is approximated by

$$s_{j+1} = \frac{f'(\mathbf{v}_j + \sigma_{j+1} \cdot h_{j+1}) - f'(\mathbf{v}_j)}{\sigma_{j+1}} \tag{5}$$

for some small σ_{j+1}. As Møller pointed out, the approximation in equation (5) is poor if the Hessian becomes indefinite, and the search might fail. Therefore, one checks whether $f'(\mathbf{v}_j)$ is positive definite or not by evaluating the sign of $\beta_{j+1} := h_{j+1}^T \cdot s_{j+1}$. If it is indefinite, i.e. $\beta_{j+1} \leq 0$, a scalar λ_{j+1} is introduced into equation (5) to regulate the indefiniteness of $f''(\mathbf{v}_j)$ by setting

$$s_{j+1} = \frac{f'(\mathbf{v}_j + \sigma_{j+1} \cdot h_{j+1}) - f'(\mathbf{v}_j)}{\sigma_{j+1}} + \lambda_{j+1} \cdot h_{j+1} \tag{6}$$

By increasing λ_{j+1} one can ensure $\beta_{j+1} > 0$. Using a simple heuristic for rising λ_{j+1}, Møller showed that the step size δ_{j+1} is given by

$$\delta_{j+1} = \frac{-h_{j+1}^T \cdot f'(\mathbf{v}_j)}{h_{j+1}^T \cdot s_{j+1} + \lambda_{j+1} \cdot \mid h_{j+1} \mid^2}$$

where s_{j+1} is computed according to (6). The values of λ_{j+1} directly scale the step size δ_{j+1} in the way, that the bigger the λ_{j+1}, the smaller the step size δ_{j+1}. The actual algorithm uses a second heuristic to adjust λ_{j+1} depending on the quality of the quadratic approximation to f in the current point. We refer to [12]. Most importantly, only the gradient needs to be computed, i.e. scaled conjugate gradient could be implemented using a Bayesian network inference engine.

Name	Description
Alarm	Well-known benchmark network for the ICU ventilator management. It consists of 37 nodes and 752 parameters [18]. No latent variables.
Insurance	Well-known benchmark network for estimating the expected claim costs for a car insurance policyholder. It consists of 27 nodes (12 latent) and over 1419 parameters [5].
3-1-3, **5-3-5**	Two artificial networks with a feed-forward architecture known from neural networks [5]. There are three fully connected layers of $3 \times 2 \times 3$ (resp. $5 \times 3 \times 5$) nodes. Nodes of the first and third layers have 3 possible states, nodes of the second 2.

Table 1: Description of the networks used in the experiments.

4 Experiments

In the experiments described below, we implemented all three algorithms using Netica API (`http://www.norsys.com`) for Bayesian network inference. We adapted the conjugate gradient (CG) described in [15] to fullfil the constraints described above. Based on this code, we adapted the scaled conjugate gradient (SCG) as implemented in Bishop and Nabney's Netlab library (`http://www.ncrg.aston.ac.uk/netlab/`, see also [17]) with an upper bound on the scale parameter of $2 \cdot 10^6$. To see how sensitive the methods to informative priors are, we also implemented each of them using BDeu priors [19] (using 1 as parameter).

4.1 Methodology

Data were generated from four Bayesian networks whose main characteristics are described in Table 1. From each target network, we generated a test set of 10000 data cases, and training sets of 100, 200, 500 and 1000 data cases with a fraction of 0, 0.1, 0.2 and 0.3 of missing at random values of the observed nodes. The values of latent nodes were never observed. For each training set, five different random initial sets of parameters were tried. We ran each algorithm on each data set starting from each of the generated initial set of parameters. We used a simple, typical stopping criterion. We stopped when a limit of 200 iteration was exceeded or a change in average log-likelihood per case was less than 10^{-5} from one iteration to the next. We chose this low threshold because EM could be very slow if high accuracy of the estimate is necessary, whereas conjugate gradients work well under these conditions. Although discrete Bayesian network are said to be robust against small changes, there exist situations where small variations can lead to significant changes in computed queries [20]. All learned model were evaluated on the test set using the *normalized-loss* $\frac{1}{M}\sum_l^M(\log P^*(\mathbf{d}_l) - \log P(\mathbf{d}_l))$ where P^* is the probability of the generating distribution. Normalized-loss measures the additional penalty for using an approximation instead of the true model, and is also a cross entropy estimate. The closer the normalized-loss is to zero, the better. We report on the average performance and average running time. To avoid a comparison of CPU times, we consider *EM-equivalent* iterations. As mentioned above, each SCG iteration involves two likelihood evaluations, i.e. two EM-equivalent iterations.

4.2 Results

The results on the normalized-loss measure are summarized in Table 2. We omit CG from further investigations because it did not terminated in reasonable time [2] on **Insurance** and **Alarm**. However, on **3-1-3** and **5-3-5**, it showed the expected behaviour. Considering only the normalized-loss, it reached normalized-losses slightly closer to zero then EM and SCG (mean difference around 0.001, priors). The number of iterations on **3-1-3** was on average twice times higher than for EM, on **5-3-5** they were on average twice lower. However, the number EM-equivalent iterations seemed to be in all cases much higher then for EM and SCG given that the overall running time was roughly four times higher. Furthermore, CG was sensible to the initial set of parameters.

[2] No run was finished after one week on a Pentium III, 450 MHz.

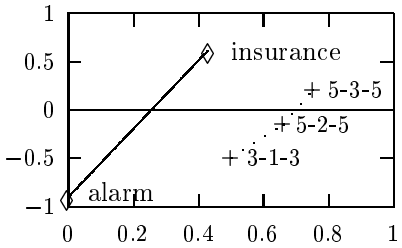

Figure 1: Percentage of latent parameters vs. speed-up in number of EM-equivalent iterations. The domains are grouped into network structures with and without encoded prior information.

Both procedures, EM and SCG reached similar normalized-losses. As expected, a higher number of data cases lead to a lower normalized-loss. Similar, informative priors lead to a lower normalized-loss. The SCG was less sensitive to informative priors than EM. This is not surprising given that EM uses the current parameters "only" to compute the expected counts, whereas gradient-based methods weight the expected counts by the current parameters. Thus, the gradient-based approaches do somehow more conservative parameter updates when propagating the initial set of parameters through the iterations. Furthermore, both were insensitive to the initial set of parameters. The EM tend to have a slightly lower variation, but a *one-tailed, paired sampled t test* shows that only a 0.045 difference in mean of EM and SCG (with informative priors, $p = 0.05$) was significant. EM reached normalized-losses slightly closer to zero in all domains but **5-3-5** where a 0.02 difference in mean was significant ($p = 0.05$) favoring SCG. However, the t test takes all restarts into account, but in practice, one usually selects for each method the best model out of the set of restarts. To compare the performance of the best models, we applied a *sign test*. It yield that the number of cases, where the best model of SCG outperformed EM's best model, was unusually larger than for EM, 43/21, but there were variations over the domains: **Insurance** 12/4, **Alarm** 9/7, **3-1-3** 6/10, and **5-3-5** 16/0.

Table 3 summarizes the number of EM-equivalent iterations. EM was significantly faster then SCG on **Alarm** . However, this did not carry over to all domains. A *one-tailed, paired sampled t test* showed that EM ran significantly faster than SCG on **Alarm**, $p = 0.00005$, and on **3-1-3**, $p = 0.0005$. The difference in mean decreased from 82 for **Alarm** to 24 for **3-1-3**. This changed for **5-3-5** and **Insurance**. Here, the same test showed that SCG ran significantly faster than EM on **5-3-5**, $p = 0.0005$, and on **Insurance**, $p = 0.00005$. The difference in mean increased from 23 for **5-3-5** to 62 for **Insurance**.

Domain / # Hidden	Method	Number of data cases			
		100	200	500	1000
Insurance	EM	$0.99 \pm 0.04 (12.20 \pm 2.03)$	$\mathbf{0.64 \pm 0.03} (9.18 \pm 0.48)$	$0.37 \pm 0.02 (2.56 \pm 0.41)$	$0.20 \pm 0.01 (0.72 \pm 0.18)$
0.0	SCG	$\mathbf{1.82 \pm 1.76} (\mathbf{8.40 \pm 0.74})$	$0.62 \pm 0.02 (\mathbf{3.92 \pm 0.76})$	$\mathbf{0.33 \pm 0.03} (\mathbf{1.33 \pm 0.29})$	$\mathbf{0.51 \pm 0.60} (\mathbf{0.56 \pm 0.19})$
0.1	EM	$1.08 \pm 0.05 (20.98 \pm 2.55)$	$0.68 \pm 0.04 (12.25 \pm 2.15)$	$0.37 \pm 0.02 (3.03 \pm 0.75)$	$\mathbf{0.23 \pm 0.01} (0.90 \pm 0.15)$
	SCG	$\mathbf{1.14 \pm 0.35} (\mathbf{8.10 \pm 1.42})$	$\mathbf{0.69 \pm 0.11} (\mathbf{4.22 \pm 1.19})$	$\mathbf{0.89 \pm 0.67} (\mathbf{1.76 \pm 0.28})$	$0.30 \pm 0.08 (\mathbf{0.76 \pm 0.04})$
0.2	EM	$\mathbf{1.07 \pm 0.03} (22.94 \pm 2.20)$	$0.76 \pm 0.03 (14.98 \pm 3.01)$	$0.45 \pm 0.02 (4.60 \pm 0.67)$	$0.25 \pm 0.02 (0.97 \pm 0.19)$
	SCG	$1.35 \pm 0.48 (\mathbf{8.85 \pm 2.62})$	$\mathbf{0.76 \pm 0.06} (\mathbf{5.80 \pm 0.29})$	$\mathbf{0.44 \pm 0.06} (\mathbf{2.64 \pm 0.49})$	$\mathbf{0.37 \pm 0.24} (\mathbf{1.06 \pm 0.19})$
0.3	EM	$1.35 \pm 0.04 (28.78 \pm 4.44)$	$0.94 \pm 0.02 (14.87 \pm 1.63)$	$\mathbf{0.52 \pm 0.03} (6.11 \pm 0.75)$	$0.29 \pm 0.01 (\mathbf{1.47 \pm 0.16})$
	SCG	$\mathbf{1.59 \pm 0.71} (\mathbf{12.26 \pm 0.75})$	$\mathbf{1.25 \pm 0.39} (\mathbf{7.04 \pm 1.17})$	$0.62 \pm 0.13 (\mathbf{3.62 \pm 0.70})$	$\mathbf{0.38 \pm 0.09} (1.43 \pm 0.08)$
Alarm	EM	$0.81 \pm 0.00 (\mathbf{-0.66 \pm 0.01})$	$\mathbf{0.62 \pm 0.00} (\mathbf{-0.44 \pm 0.02})$	$\mathbf{0.29 \pm 0.00} (\mathbf{-0.15 \pm 0.01})$	$0.16 \pm 0.00 (\mathbf{-0.03 \pm 0.00})$
0.0	SCG	$\mathbf{1.01 \pm 0.25} (3.79 \pm 0.18)$	$0.94 \pm 0.37 (2.41 \pm 0.20)$	$0.33 \pm 0.02 (1.01 \pm 0.05)$	$\mathbf{0.17 \pm 0.00} (0.53 \pm 0.04)$
0.1	EM	$1.28 \pm 0.00 (13.34 \pm 1.38)$	$0.68 \pm 0.00 (7.01 \pm 0.87)$	$0.31 \pm 0.00 (3.30 \pm 0.08)$	$0.18 \pm 0.00 (1.42 \pm 0.07)$
	SCG	$\mathbf{1.81 \pm 0.74} (\mathbf{5.94 \pm 0.21})$	$\mathbf{0.87 \pm 0.27} (\mathbf{2.53 \pm 0.13})$	$\mathbf{0.31 \pm 0.00} (\mathbf{1.04 \pm 0.05})$	$\mathbf{0.19 \pm 0.00} (\mathbf{0.54 \pm 0.02})$
0.2	EM	$\mathbf{1.10 \pm 0.00} (18.47 \pm 1.43)$	$0.64 \pm 0.00 (8.91 \pm 0.76)$	$0.31 \pm 0.00 (2.75 \pm 0.43)$	$\mathbf{0.24 \pm 0.00} (1.61 \pm 0.06)$
	SCG	$1.31 \pm 0.19 (\mathbf{6.06 \pm 0.26})$	$\mathbf{1.25 \pm 0.80} (\mathbf{3.18 \pm 0.09})$	$\mathbf{0.31 \pm 0.00} (\mathbf{1.09 \pm 0.07})$	$0.24 \pm 0.01 (\mathbf{0.64 \pm 0.03})$
0.3	EM	$1.22 \pm 0.00 (16.63 \pm 1.31)$	$\mathbf{0.65 \pm 0.00} (11.41 \pm 0.79)$	$\mathbf{0.41 \pm 0.00} (3.98 \pm 0.13)$	$\mathbf{0.21 \pm 0.00} (1.68 \pm 0.05)$
	SCG	$\mathbf{1.28 \pm 0.08} (\mathbf{6.76 \pm 0.40})$	$0.66 \pm 0.01 (\mathbf{4.07 \pm 0.27})$	$1.12 \pm 0.96 (\mathbf{1.70 \pm 0.09})$	$0.22 \pm 0.00 (\mathbf{0.76 \pm 0.04})$
3-1-3	EM	$\mathbf{0.15 \pm 0.01} (\mathbf{0.30 \pm 0.26})$	$0.07 \pm 0.00 (0.12 \pm 0.00)$	$\mathbf{0.03 \pm 0.00} (0.03 \pm 0.00)$	$0.02 \pm 0.00 (0.02 \pm 0.00)$
0.0	SCG	$0.16 \pm 0.02 (0.57 \pm 0.04)$	$\mathbf{0.07 \pm 0.00} (\mathbf{0.12 \pm 0.03})$	$0.03 \pm 0.00 (\mathbf{0.03 \pm 0.00})$	$\mathbf{0.02 \pm 0.00} (\mathbf{0.02 \pm 0.00})$
	CG	$0.17 \pm 0.04 (0.56 \pm 0.08)$	$0.07 \pm 0.00 (0.10 \pm 0.00)$	$0.04 \pm 0.02 (0.03 \pm 0.00)$	$0.02 \pm 0.00 (0.02 \pm 0.00)$
0.1	EM	$0.13 \pm 0.02 (\mathbf{0.46 \pm 0.35})$	$0.13 \pm 0.03 (\mathbf{0.18 \pm 0.02})$	$0.05 \pm 0.00 (\mathbf{0.06 \pm 0.00})$	$\mathbf{0.02 \pm 0.00} (\mathbf{0.02 \pm 0.00})$
	SCG	$\mathbf{0.13 \pm 0.01} (0.31 \pm 0.05)$	$\mathbf{0.11 \pm 0.01} (0.17 \pm 0.02)$	$\mathbf{0.06 \pm 0.03} (0.06 \pm 0.00)$	$0.02 \pm 0.00 (0.02 \pm 0.00)$
	CG	$0.13 \pm 0.02 (0.31 \pm 0.04)$	$0.11 \pm 0.02 (0.17 \pm 0.02)$	$0.05 \pm 0.00 (0.06 \pm 0.00)$	$0.02 \pm 0.01 (0.02 \pm 0.00)$
0.2	EM	$\mathbf{0.20 \pm 0.01} (2.08 \pm 0.90)$	$\mathbf{0.16 \pm 0.04} (0.28 \pm 0.08)$	$\mathbf{0.05 \pm 0.00} (\mathbf{0.07 \pm 0.01})$	$\mathbf{0.03 \pm 0.00} (\mathbf{0.04 \pm 0.00})$
	SCG	$0.25 \pm 0.05 (\mathbf{0.86 \pm 0.31})$	$0.10 \pm 0.01 (\mathbf{0.20 \pm 0.08})$	$0.05 \pm 0.00 (0.11 \pm 0.03)$	$0.04 \pm 0.03 (0.04 \pm 0.00)$
	CG	$0.20 \pm 0.03 (0.79 \pm 0.26)$	$0.10 \pm 0.01 (0.24 \pm 0.09)$	$0.06 \pm 0.00 (0.11 \pm 0.02)$	$0.03 \pm 0.00 (0.04 \pm 0.00)$
0.3	EM	$\mathbf{0.15 \pm 0.00} (1.47 \pm 0.18)$	$\mathbf{0.16 \pm 0.01} (0.90 \pm 0.16)$	$\mathbf{0.03 \pm 0.00} (0.06 \pm 0.00)$	$0.05 \pm 0.05 (0.06 \pm 0.05)$
	SCG	$0.17 \pm 0.02 (\mathbf{0.77 \pm 0.28})$	$0.17 \pm 0.01 (\mathbf{0.68 \pm 0.22})$	$0.04 \pm 0.03 (\mathbf{0.06 \pm 0.00})$	$\mathbf{0.02 \pm 0.00} (\mathbf{0.02 \pm 0.00})$
	CG	$0.16 \pm 0.02 (0.79 \pm 0.31)$	$0.18 \pm 0.04 (0.72 \pm 0.16)$	$0.03 \pm 0.00 (0.06 \pm 0.00)$	$0.02 \pm 0.00 (0.03 \pm 0.00)$
5-3-5	EM	$0.23 \pm 0.01 (17.36 \pm 0.92)$	$0.22 \pm 0.01 (8.38 \pm 1.23)$	$0.15 \pm 0.01 (1.86 \pm 0.41)$	$0.11 \pm 0.00 (0.43 \pm 0.08)$
0.0	SCG	$\mathbf{0.19 \pm 0.03} (\mathbf{6.00 \pm 0.88})$	$\mathbf{0.21 \pm 0.01} (\mathbf{2.96 \pm 1.32})$	$\mathbf{0.13 \pm 0.02} (\mathbf{0.94 \pm 0.09})$	$\mathbf{0.10 \pm 0.01} (\mathbf{0.28 \pm 0.02})$
	CG	$0.14 \pm 0.07 (5.15 \pm 0.84)$	$0.13 \pm 0.07 (3.28 \pm 0.61)$	$0.09 \pm 0.05 (0.84 \pm 0.12)$	$0.06 \pm 0.03 (0.29 \pm 0.02)$
0.1	EM	$0.19 \pm 0.01 (19.37 \pm 2.14)$	$0.21 \pm 0.01 (11.76 \pm 2.05)$	$0.18 \pm 0.00 (3.03 \pm 0.34)$	$0.13 \pm 0.01 (1.03 \pm 0.22)$
	SCG	$\mathbf{0.14 \pm 0.04} (\mathbf{7.52 \pm 1.59})$	$\mathbf{0.20 \pm 0.01} (\mathbf{4.43 \pm 0.66})$	$\mathbf{0.15 \pm 0.02} (\mathbf{1.47 \pm 0.34})$	$\mathbf{0.09 \pm 0.02} (\mathbf{0.47 \pm 0.05})$
	CG	$0.12 \pm 0.06 (6.50 \pm 0.65)$	$0.13 \pm 0.07 (4.08 \pm 0.35)$	$0.11 \pm 0.06 (1.34 \pm 0.07)$	$0.08 \pm 0.04 (0.44 \pm 0.06)$
0.2	EM	$0.16 \pm 0.01 (24.81 \pm 1.67)$	$0.18 \pm 0.01 (13.88 \pm 2.74)$	$0.16 \pm 0.01 (4.09 \pm 0.63)$	$0.14 \pm 0.01 (1.04 \pm 0.16)$
	SCG	$\mathbf{0.12 \pm 0.03} (\mathbf{7.64 \pm 0.93})$	$\mathbf{0.14 \pm 0.03} (\mathbf{5.58 \pm 0.66})$	$\mathbf{0.14 \pm 0.02} (\mathbf{1.56 \pm 0.61})$	$\mathbf{0.13 \pm 0.01} (\mathbf{0.72 \pm 0.07})$
	CG	$0.09 \pm 0.05 (8.08 \pm 1.00)$	$0.10 \pm 0.05 (4.50 \pm 0.50)$	$0.09 \pm 0.05 (1.62 \pm 0.24)$	$0.08 \pm 0.04 (0.72 \pm 0.09)$
0.3	EM	$0.25 \pm 0.01 (26.53 \pm 2.76)$	$0.17 \pm 0.00 (13.59 \pm 2.25)$	$0.25 \pm 0.01 (4.88 \pm 0.93)$	$0.15 \pm 0.00 (1.59 \pm 0.30)$
	SCG	$\mathbf{0.20 \pm 0.04} (\mathbf{8.32 \pm 1.15})$	$\mathbf{0.16 \pm 0.01} (\mathbf{5.53 \pm 0.70})$	$\mathbf{0.21 \pm 0.03} (\mathbf{2.68 \pm 0.15})$	$\mathbf{0.13 \pm 0.02} (\mathbf{0.95 \pm 0.19})$
	CG	$0.15 \pm 0.08 (7.99 \pm 0.72)$	$0.10 \pm 0.05 (4.45 \pm 0.26)$	$0.15 \pm 0.08 (2.68 \pm 0.45)$	$0.09 \pm 0.05 (0.92 \pm 0.12)$

Table 2: Average normalized-loss of **Insurance**, **Alarm**, **3-1-3**, and **5-3-5** on an independent test set consisting of 10000 data cases. The mean and standard deviation of five restarts are reported (bracket term: without BDeu priors). A bold term indicates the method which estimated the best model (CG not considered). When there is no value for conjugate gradient (CG) this indicates that one run of it took longer than one week on this domain.

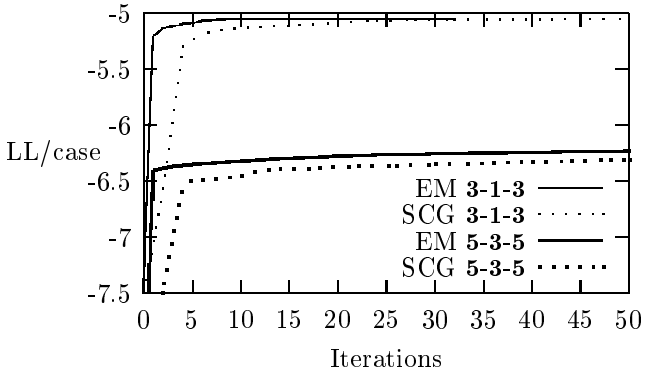

Figure 2: Learning curves averaged over five restarts for 1000 data cases, no missing values.

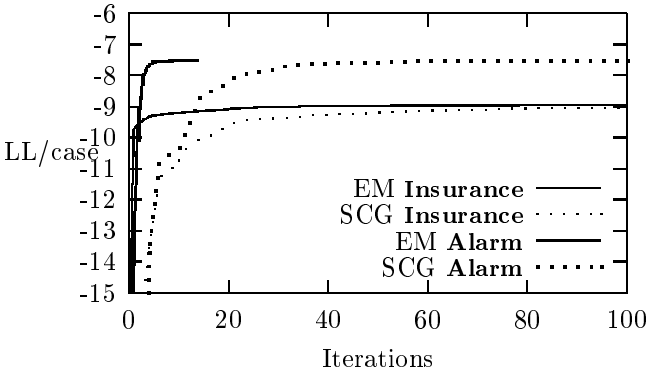

Figure 3: Learning curves averaged over five restarts for 1000 data cases. Fraction of 0.3 missing values for **Alarm**, no missing values for **Insurance**.

To summarize, an acceleration of conjugate gradients is possible. SCG seems to have some EM-like characteristics. They are fast, and insensitive to the initial set of parameters. This is remarkable given that conjugate gradients are usually consider to perform worth than EM. But the behaviors of both algorithms varied over the domain, and we can identify "easy" and "hard" domains for both algorithms. **Alarm** and **Insurance** are networks where prior knowledge is encoded into the structure, whereas **3-1-3** and **5-3-5** have uninformative structures. Within these types of network structures, the fraction of missing values and therefore the number latent parameters seems to correlate to hard domains. Figure 1 shows the percentage of latent parameters vs. speed-up which is defined as

Domain / # Hidden	Method	Number of data cases			
		100	200	500	1000
Insurance	EM	$100 \pm 28 (144 \pm 34)$	$143 \pm 40 (199 \pm 0)$	$188 \pm 17 (199 \pm 0)$	$188 \pm 13 (199 \pm 0)$
0.0	SCG	$95 \pm 40 (307 \pm 75)$	$132 \pm 24 (272 \pm 67)$	$138 \pm 42 (380 \pm 34)$	$156 \pm 82 (335 \pm 130)$
0.1	EM	$129 \pm 37 (128 \pm 32)$	$178 \pm 19 (193 \pm 9)$	$168 \pm 25 (199 \pm 0)$	$177 \pm 27 (199 \pm 0)$
	SCG	$42 \pm 14 (280 \pm 36)$	$136 \pm 58 (302 \pm 74)$	$91 \pm 67 (318 \pm 43)$	$89 \pm 54 (355 \pm 42)$
0.2	EM	$171 \pm 26 (153 \pm 29)$	$161 \pm 42 (199 \pm 0)$	$196 \pm 6 (199 \pm 0)$	$178 \pm 19 (199 \pm 0)$
	SCG	$64 \pm 25 (251 \pm 68)$	$141 \pm 47 (357 \pm 67)$	$112 \pm 42 (335 \pm 44)$	$106 \pm 71 (377 \pm 29)$
0.3	EM	$132 \pm 38 (180 \pm 20)$	$156 \pm 27 (199 \pm 0)$	$187 \pm 13 (199 \pm 0)$	$180 \pm 36 (199 \pm 0)$
	SCG	$77 \pm 41 (272 \pm 69)$	$79 \pm 43 (328 \pm 49)$	$89 \pm 57 (312 \pm 59)$	$88 \pm 51 (312 \pm 50)$
Alarm	EM	$1 \pm 0 (1 \pm 0)$	$1 \pm 0 (1 \pm 0)$	$1 \pm 0 (1 \pm 0)$	$1 \pm 0 (1 \pm 0)$
0.0	SCG	$66 \pm 18 (97 \pm 12)$	$75 \pm 46 (114 \pm 13)$	$104 \pm 20 (148 \pm 15)$	$130 \pm 31 (162 \pm 14)$
0.1	EM	$7 \pm 0 (10 \pm 1)$	$8 \pm 0 (13 \pm 3)$	$6 \pm 0 (9 \pm 0)$	$6 \pm 0 (8 \pm 0)$
	SCG	$83 \pm 36 (130 \pm 8)$	$83 \pm 46 (131 \pm 17)$	$140 \pm 18 (147 \pm 12)$	$113 \pm 14 (158 \pm 9)$
0.2	EM	$11 \pm 0 (17 \pm 2)$	$11 \pm 0 (16 \pm 3)$	$10 \pm 2 (13 \pm 2)$	$9 \pm 0 (12 \pm 1)$
	SCG	$84 \pm 22 (167 \pm 26)$	$87 \pm 13 (174 \pm 27)$	$100 \pm 7 (170 \pm 32)$	$120 \pm 26 (142 \pm 9)$
0.3	EM	$16 \pm 1 (25 \pm 2)$	$14 \pm 0 (23 \pm 2)$	$15 \pm 1 (18 \pm 1)$	$13 \pm 1 (17 \pm 1)$
	SCG	$83 \pm 27 (163 \pm 22)$	$108 \pm 26 (173 \pm 7)$	$69 \pm 45 (164 \pm 6)$	$106 \pm 10 (174 \pm 11)$
3-1-3	EM	$21 \pm 10 (27 \pm 7)$	$21 \pm 2 (22 \pm 2)$	$18 \pm 3 (22 \pm 4)$	$16 \pm 1 (18 \pm 1)$
0.0	SCG	$49 \pm 24 (58 \pm 15)$	$63 \pm 6 (72 \pm 30)$	$66 \pm 15 (83 \pm 6)$	$60 \pm 7 (67 \pm 14)$
0.1	EM	$28 \pm 4 (46 \pm 13)$	$26 \pm 7 (35 \pm 17)$	$27 \pm 7 (36 \pm 8)$	$23 \pm 7 (27 \pm 8)$
	SCG	$54 \pm 11 (76 \pm 11)$	$57 \pm 13 (72 \pm 35)$	$44 \pm 8 (87 \pm 15)$	$62 \pm 5 (76 \pm 8)$
0.2	EM	$24 \pm 9 (36 \pm 11)$	$52 \pm 36 (43 \pm 14)$	$30 \pm 6 (36 \pm 6)$	$27 \pm 6 (28 \pm 7)$
	SCG	$54 \pm 23 (58 \pm 15)$	$60 \pm 7 (89 \pm 10)$	$51 \pm 10 (61 \pm 20)$	$54 \pm 17 (80 \pm 11)$
0.3	EM	$37 \pm 9 (40 \pm 8)$	$46 \pm 8 (62 \pm 18)$	$32 \pm 3 (36 \pm 3)$	$35 \pm 14 (40 \pm 17)$
	SCG	$55 \pm 36 (70 \pm 10)$	$68 \pm 26 (78 \pm 12)$	$57 \pm 23 (80 \pm 12)$	$57 \pm 12 (74 \pm 16)$
5-3-5	EM	$97 \pm 7 (38 \pm 6)$	$92 \pm 40 (52 \pm 13)$	$120 \pm 39 (78 \pm 17)$	$113 \pm 24 (85 \pm 7)$
0.0	SCG	$75 \pm 37 (151 \pm 60)$	$134 \pm 33 (108 \pm 48)$	$151 \pm 50 (166 \pm 7)$	$150 \pm 37 (142 \pm 6)$
0.1	EM	$160 \pm 12 (56 \pm 16)$	$116 \pm 19 (67 \pm 14)$	$142 \pm 25 (84 \pm 11)$	$162 \pm 14 (126 \pm 20)$
	SCG	$80 \pm 48 (207 \pm 45)$	$130 \pm 13 (184 \pm 34)$	$112 \pm 22 (141 \pm 21)$	$99 \pm 35 (158 \pm 19)$
0.2	EM	$128 \pm 35 (72 \pm 17)$	$190 \pm 19 (78 \pm 16)$	$160 \pm 10 (105 \pm 20)$	$141 \pm 10 (100 \pm 19)$
	SCG	$99 \pm 62 (180 \pm 38)$	$97 \pm 37 (168 \pm 25)$	$150 \pm 58 (140 \pm 37)$	$114 \pm 23 (163 \pm 25)$
0.3	EM	$166 \pm 33 (65 \pm 7)$	$99 \pm 15 (78 \pm 16)$	$142 \pm 29 (112 \pm 22)$	$157 \pm 23 (131 \pm 17)$
	SCG	$72 \pm 42 (170 \pm 21)$	$88 \pm 20 (179 \pm 29)$	$121 \pm 28 (151 \pm 9)$	$130 \pm 26 (143 \pm 22)$

Table 3: Average EM equivalent iterations performed by EM and SCG to learn **Insurance**,**Alarm**, **3-1-3**, and **5-3-5**. The mean and standard deviation of five restarts are reported (bracket term: without BDeu priors). An entry of 199 ± 0 for EM indicates that it exceeded the maximum number of iteration in all restarts.

the quotient of average EM-equivalent iterations (over corresponding restarts) of EM and SCG minus one. To test our hypothesis, we investigated the network **5-2-5**. It is a variant of **5-3-5** with only two latent variables. The observable variables have two, the latent one three states. We tested **5-2-5** with the same experiments as before but restricted to 1000 data cases. The speed-up on **5-2-5** was -0.15, i.e. between the speed-up of **3-1-3** and **5-3-5**. Although this is preliminary, the result is promising.

Finally, as Figures 2 and 3 show, EM is usually faster when far away from the solution. Therefore, the EM seems to be the domain independent method of choice. But, the results show that an acceleration of conjugate gradients is possible. SCG can be superior to EM, and in contrast to CG, it learned **Insurance** and **Alarm**.

5 Related Work

Both, EM and gradient-based algorithms are well-known parameter estimation techniques. For a general introduction we refer to [10, 13]. Bishop [17] and Møller [12] show how to use (scaled) conjugate-gradients to estimate parameters of neural networks. Lauritzen [4] introduced EM for Bayesian networks (see also Heckerman [2] for a nice tutorial). The original work on gradient-based approaches in the context of Bayesian networks was done by Binder *et al.* [5], some recent work by Jensen [6, 7]. However, we are neither aware of any direct experimental comparison between the two approaches nor of research investigating accelerated conjugate gradients such as scaled conjugate gradients. Binder *et al.* [5] used traditional conjugate gradient ascent in their experiments. They compared its performance on networks where prior knowledge was encoded into the structure and networks where this was not done. They did not compare different parameter estimation approaches. Bauer *et al.* [11] reported on experiments with different learning algorithms for Bayesian networks, but they focused on accelerations of the EM. They did not report on results of conjugate gradients. Thiesson [8] discussed (traditional) conjugate gradient accelerations of the EM, but does not report on experiments. Ortiz and Kaelbling [9] conducted experiments with several learning algorithms (mainly accelerated EMs) in the context of continuous models, namely density estimation with a mixture of Gaussian. They argued "that conjugate gradient by itself is not a good idea since it requires a very precise line search when it is far away from a solution". Our experiments do not support this for discrete Bayesian networks. Ortiz and Kaelbling also gave a classification of their domains into "easy" and "hard" ones. Like in our case, accelerated methods improved convergence on hard problems. However, they accelerated EM and not conjugate gradient. Furthermore, they suggested a comparison in the context of discrete Bayesian networks which we have provided here.

6 Conclusions and future work

In this case study, we experimentally compared two classes of maximum likelihood parameter estimators, EM and gradient-based algorithms, on several Bayesian networks. To overcome the expensive line search of traditional conjugate gradients, we applied scaled conjugate gradients. This type of accelerated conjugate gradient is novel in the context of Bayesian networks. The experiments show that accelerating conjugate gradients is possible. They are superior to traditional gradients. Furthermore, they are competetive with EM. In general EM seems to be the domain independent method of choice, but which technique is superior depends on properties of the domain. We identified "easy" and "hard" instances for both algorithms. Scaled conjugate gradients seem especially well suited to learn problem instances with many latent parameters. Given that gradients are easily applied to other scoring functions than likelihood, accelerated conjugate gradients are a promising alternative to EM.

Our results suggest that the number of latent variables determines hard instances. In the future, we will conduct more experiments to validate this and to find other properties. There have been several proposals of accelerating EM using conjugate gradients. Comparing these algorithms with scaled conjugate gradients should yield further Another interesting direction of future research is to apply a stochastic scaled conjugate gradient (see [21]) to the problem of *online-learning* of Bayesian networks.

Acknowledgements

The authors would like to thank Luc De Raedt and Wolfram Burgard for helpful comments on the paper. Kristian Kersting was supported by the European Union IST programme under contract number IST-2001-33053 (Application of Probabilistic Inductive Logic Programming - APRIL).

References

[1] J. Pearl. *Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2. edition, 1991.

[2] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, March 1995.

[3] A. Dempster, N. Larid, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[4] S. L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.

[5] J.Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, pages 213–244, 1997.

[6] F. V. Jensen. Gradient descent training of bayesian networks. In *Proceedings of the Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU-99)*, 1999.

[7] F. V. Jensen. *Bayesian networks and decision graphs*. Springer-Verlag New York, 2001.

[8] B. Thiesson. Accelerated quantification of Bayesian networks with incomplete data. In *Proceedings of First International Conference on Knowledge Discovery and Data Mining*, pages 306–311, 1995.

[9] L. E. Ortiz and L. Pack Kaelbling. Accelerating EM: An Empirical Study. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Articial Intelligence (UAI-99)*, pages 512–521, 1999.

[10] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, New York, 1997.

[11] E. Bauer, D. Koller, and Y. Singer. Update Rules for Parameter Estimation in Bayesian Networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, 1997.

[12] M. Møller. A Scaled Conjugate Gradient Algoritm for Fast Supervised Learning. *Neural Networks*, 6:525–533, 1993.

[13] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 2. edition, 1984.

[14] J. Nocedal. Theory of algorithms for unconstrained optimization. *Acta Numerica*, pages 199–242, 1992.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 2. edition, 1993.

[16] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[17] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[18] I. Beinlinch, H. Suermondt, R. Chavez, and G. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the Second European Conference on Articial Intelligence in Medicine*, pages 247–256, 1989.

[19] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[20] H. Chan and A. Darwiche. When do numbers really matter? In *Proceedings of the Seventeenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*.

[21] M. Møller. Supervised Learning on Large Redundant Training sets. *International Journal Neural Systems*, 4(1):15–25, 1993.