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

Kristian Kersting[1] and Niels Landwehr[1]

Institute for Computer Science, Machine Learning Lab
Albert-Ludwigs-University, Georges-Köhler-Allee, Gebäude 079,
D-79085 Freiburg i. Brg., Germany

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. We will 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

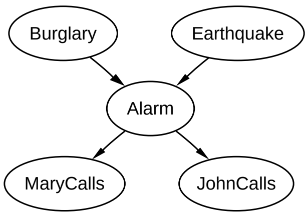
Bayesian networks are one of the most important, efficient and elegant frameworks for representing and reasoning with probabilistic models. They specify joint probability distributions over finite sets of random variables, and have been applied to many real-world problems in diagnosis, forecasting, automated vision, sensor fusion and manufacturing control. Over the past years, there has been much interest in the problem of learning Bayesian networks from data to avoid the problems of knowledge elicitation. For learning Bayesian networks, *parameter estimation* is a fundamental task not only because of the inability of humans to reliably estimate the parameters, but also because it forms the basis for the overall learning problem [10].

A classical method for parameter estimation is *maximum likelihood* parameter estimation. Here, one selects those parameters maximizing the likelihood which is the probability of the observed data as a function of the unknown parameters with respect to the current model. Unfortunately in many real-world domains, the data cases available are incomplete, i.e. some values may not be observed. For instance in medical domains, a patient rarely gets all of the possible tests. In presence of missing data, the parameter estimation becomes much more difficult [6]. In fact, 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 latter ones have been proven to be NP-hard [7]. The most prominent techniques within Bayesian networks are the Expectation-Maximization (EM) algorithm and gradient-based approaches.

The comparison between EM and (advanced) gradient techniques 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 easily allow e.g. to learn non-multinomial parameterizations using the chain rule for derivatives [3] or to choose other scoring functions than the likelihood [13]. The EM algorithm may slowly converge when close to a solution, but can be sped up near the maximum using gradient-based approaches [26,17,1,23]. Other acceleration approaches execute only a partial maximization step of the EM algorithm based on gradient techniques leading to generalized EM algorithms [17].

In this context, we report on an experimental comparison between EM and (accelerated) conjugate gradients to maximum likelihood parameter estimation within Bayesian networks over finite random variables. In doing so, we follow Ortiz and Kaelbling's [23] suggestion for such a comparison in the context of (discrete) Bayesian network. To overcome the expensive line search, we adapted *scaled conjugate gradients* [18] from the field of learning neural networks. They avoid the line search by using a Levenberg-Marquardt approach [16] in order to scale the step size, and employ the Hessian of the scoring function to quadratically extrapolate the maximum instead of doing a line search. This type of accelerated conjugate gradients are novel in the context of Bayesian networks. Other work investigated approximated line searches only to accelerate EM [26,23,1]. From the experimental results, we will argue that scaled conjugate gradients are competitive with EM. (Accelerated) gradient methods can be superior depending on properties of the domain such as the fraction of latent parameters and the prior information encoded in the network structure, though EM seems to be the best *domain independent* choice.

We proceed as follows. After briefly introducing Bayesian networks in Section 2, we review maximum likelihood estimation via the EM algorithm and simple gradient-ascent in Section 3. In Section 4, we briefly review a well-known criterion to classify data sets into "EM-hard" and "EM-easy" ones. Section 5 reviews conjugate gradients and introduces scaled conjugate gradients. In Section 6, we experimentally compare the EM algorithm with



Burglary	*Earthquake*	$\mathbf{P}(Alarm)$
true	*true*	$(0.95, 0.05)$
true	*false*	$(0.94, 0.06)$
false	*true*	$(0.29, 0.71)$
false	*false*	$(0.001, 0.999)$

Fig. 1. An example of a Bayesian network modelling a burglary alarm system

conjugate gradients and scaled conjugate gradients. Before concluding, we discuss related work.

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). We will use $\mathbf{P}$ to denote a probability distribution, e.g. $\mathbf{P}(X)$, and P to denote a probability value, e.g. $P(x)$.

A *Bayesian network* [24] represents the joint probability distribution $\mathbf{P}(\mathbf{X})$ over a set $\mathbf{X} = \{X_1, \ldots, X_n\}$ of random variables. In this paper, we restrict each X_i to have a finite set $x_i^1, \ldots, x_i^{k_i}$ of possible states. Consider Judea Pearl's famous *Alarm* network graphically illustrated in Fig. 1. The (boolean) random variables are *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*. A Bayesian network is an augmented, acyclic graph, where each node corresponds to a variable X_i and each edge indicates a direct influence among the variables. For instance, *Burglary* and *Earthquake* directly affects the probability of the *Alarm* going off. We denote the parents of X_i in a graph-theoretical sense by $\mathbf{Pa}_i$, e.g. $\mathbf{Pa}_{Alarm} = \{Burglary, Earthquake\}$. The family of X_i is $\mathbf{Fa}_i := \{X_i\} \cup \mathbf{Pa}_i$, e.g. $\mathbf{Fa}_{Alarm} = \{Alarm, Burglary, Earthquake\}$. With each node X_i, a conditional probability table is associated specifying the distribution $\mathbf{P}(X_i \mid \mathbf{Pa}_i)$, cf. Fig. 1. The table entries are

$$\theta_{ijk} = P(X_i = x_i^k \mid \mathbf{Pa}_i = \mathbf{pa}_i^j) \ , \tag{1}$$

where $\mathbf{pa}_i^j$ denotes the jth joint state of the X_i's parents, e.g. $\mathbf{pa}_{Alarm}^2 = \{Burglary = true, Earthquake = false\}$. 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

$$\mathbf{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbf{P}(X_i \mid \mathbf{Pa}_i) \ . \tag{2}$$

In the rest of the paper, we will represent a Bayesian network with given structure by the vector $\boldsymbol{\theta}$ consisting of all θ_{ijk}'s. For instance for the *alarm* network we have $\boldsymbol{\theta} = (\ldots, 0.95, 0.05, 0.94, 0.06, 0.29, 0.71, 0.001, 0.999, \ldots)$.

3 Basic Maximum Likelihood Estimation

Our task is to learn the numerical parameters θ_{ijk} for a Bayesian network of a given structure. More formally, we have some initial model (assignment of parameters) $\boldsymbol{\theta}$. We also have some set of data cases $\mathbf{D} = \{\mathbf{d}_1, \ldots, \mathbf{d}_N\}$. Each data case $\mathbf{d}_i$ is a (possibly) partial assignment of values to variables in the network, e.g. for the *alarm* network

ID	*Burglary*	*Earthquake*	*Alarm*	*JohnCalls*	*MaryCalls*
$\mathbf{d}_1$	*true*	*false*	*true*	*true*	*false*
$\mathbf{d}_2$	*false*	*?*	*true*	*false*	*true*
...	...	...	...	...	...

where the question mark denotes unobserved states. We assume that the data cases are independently sampled from identical distributions (i.i.d). We want to construct the model $\boldsymbol{\theta}^*$ which best explains the data $\mathbf{D}$.

We quantify 'which best explains the data', as usual, via the likelihood of the data $\mathbf{D}$ in $\boldsymbol{\theta}$. The likelihood $L(\mathbf{D}, \boldsymbol{\theta})$ is the probability of the data $\mathbf{D}$ as a function of the unknown parameters $\boldsymbol{\theta}$, i.e.

$$L(\mathbf{D}, \boldsymbol{\theta}) = P(\mathbf{D} \mid \boldsymbol{\theta}) \ . \tag{3}$$

It simplifies because of the i.i.d. assumption to

$$L(\mathbf{D}, \boldsymbol{\theta}) = \prod_{l=1}^{N} P(\mathbf{d}_l \mid \boldsymbol{\theta}) \ . \tag{4}$$

Thus, the search space $\mathcal{H}$ to be explored is spanned by the space over the possible values of $\boldsymbol{\theta}$. We seek to find that $\boldsymbol{\theta}^* \in \mathcal{H}$ that maximizes the likelihood, i.e., $\boldsymbol{\theta}^* = \mathrm{argmax}_{\boldsymbol{\theta}\in\mathcal{H}}\, L(\mathbf{D}, \boldsymbol{\theta})$. Due to the monotonicity of the logarithm, this simplifies to $\boldsymbol{\theta}^* = \mathrm{argmax}_{\boldsymbol{\theta}\in\mathcal{H}} \log L(\mathbf{D}, \boldsymbol{\theta}) = \mathrm{argmax}_{\boldsymbol{\theta}\in\mathcal{H}} \sum_{l=1}^{N} \log P(\mathbf{d}_l \mid \boldsymbol{\theta})$, and , finally, due to the monotonicity of the logarithm, this yields

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\mathcal{H}}{\mathrm{argmax}} \sum_{l=1}^{N} \log P(\mathbf{d}_l \mid \boldsymbol{\theta}) \ . \tag{5}$$

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 as in data case $\mathbf{d}_1$, Lauritzen [15] showed that maximum likelihood estimation simply corresponds to frequency counting in the following way. Let $\mathrm{c}(\mathbf{a} \mid \mathbf{D})$ denote the *counts* for a particular joint state $\mathbf{a}$ of variables $\mathbf{A}$ in the data cases, i.e.

the number of cases in which the variables in $\mathbf{A}$ are assigned the values $\mathbf{a}$, then

$$\theta^*_{ijk} = \frac{\mathrm{c}(\mathbf{fa}_i^{jk} \mid \mathbf{D})}{\mathrm{c}(\mathbf{pa}_i^j \mid \mathbf{D})} \,, \tag{6}$$

where $\mathbf{fa}_i^{jk}$ denotes the joint state consisting of the jth state of variable X_i and the kth joint state of $\mathbf{Pa}_i$, e.g. $\mathbf{fa}^{2,3}_{Alarm} = \{\mathit{Alarm} = \mathit{false}, \mathit{Burglary} = \mathit{false}, \mathit{Earthquake} = \mathit{true}\}$. However, in the presence of missing data as in data case $\mathbf{d}_2$, the maximum likelihood estimates typically cannot be written in closed form, and iterative optimization schemes like the EM or conjugate gradient algorithms are needed. Both approaches will be introduced in turn in the following two subsections.

Expectation-Maximization

The Expectation-Maximization algorithm [8] is a classical approach to realize maximum likelihood estimation in the presence of missing values. The basic observation underlying the Expectation-Maximization algorithm is the observation of the last section: learning would be easy if we would knew the values for all the random variables. Therefore, it iteratively performs two steps to find the maximum likelihood parameters of a model:

1. Based on the current parameters $\boldsymbol{\theta}$ and the observed data $\mathbf{D}$, the algorithm computes a distribution over all possible completions of each partially observed data case.
2. Each completion is then treated as a fully-observed data case weighted by its probability. New parameters are then computed based on (6).

Lauritzen [15] showed that this idea leads to a modified (6) where *expected counts* ec instead of *counts* c are used:

$$\theta^*_{ijk} = \frac{\mathrm{ec}(\mathbf{fa}_i^{jk} \mid \mathbf{D})}{\mathrm{ec}(\mathbf{pa}_i^j \mid \mathbf{D})} \,. \tag{7}$$

Expected counts are defined as follows $\mathrm{ec}(\mathbf{a}) = \sum_{l=1}^{N} P(\mathbf{a} \mid \mathbf{d}_l, \boldsymbol{\theta}^n)$. To compute $P(\mathbf{a} \mid \mathbf{d}_l, \boldsymbol{\theta}^n)$ any Bayesian network inference engine can be used.

Gradient-Ascent Learning

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

1. It computes the *gradient* vector $\nabla_{\boldsymbol{\theta}}$ of partial derivatives of the scoring function with respect to the parameters of a Bayesian network at a given point $\boldsymbol{\theta} \in \mathcal{H}$.
2. Then it takes a small step in the direction of the gradient to the point $\boldsymbol{\theta} + \delta\nabla_{\boldsymbol{\theta}}$ where δ is the step-size parameter.

The algorithm will converge to a local maximum for small enough δ (cf. [16]). Thus, we have to compute the partial derivatives of $\log P(\mathbf{D} \mid \boldsymbol{\theta})$ with respect to parameters θ_{ijk}. According to Binder *et al.* [3], they are given by

$$\frac{\partial \log P(\mathbf{D} \mid \boldsymbol{\theta})}{\partial \theta_{ijk}} = \sum_{l=1}^{N} \frac{P(\mathbf{fa}_i^{jk} \mid \mathbf{d}_l, \boldsymbol{\theta})}{\theta_{ijk}} = \frac{\mathrm{ec}(\mathbf{fa}_i^{jk} \mid \mathbf{D})}{\theta_{ijk}} \,. \tag{8}$$

Like for the EM algorithm, any Bayesian network inference engine can be used to compute $P(\mathbf{fa}_i^{jk} \mid \mathbf{d}_l, \boldsymbol{\theta})$. However in contrast to the EM algorithm, the described gradient-ascent method has to be modified to take into account the constraint that the parameter vector $\boldsymbol{\theta}$ consists of probability values, i.e. $\theta_{ijk} \in [0,1]$ and $\sum_j \theta_{ijk} = 1$. As Binder *et al.* explain, there are two ways to enforce this:

1. Projecting the gradient onto the constraint surface, and
2. reparameterizing the problem so that the new parameters automatically respect the constraints on θ_{ijk} no matter what their values are.

We choose the latter approach as the reparameterized problem is fully unconstrained. More precisely, we define the parameters $\beta_{ijk} \in \mathbb{R}$ such that

$$\theta_{ijk} = \frac{\beta_{ijk}^2}{\sum_l \beta_{ilk}^2} \,. \tag{9}$$

where the β_{ijk} are indexed like θ_{ijk}. This enforces the constraints given above, and a local maximum with respect to β_{ijk} is also a local maximum with respect to θ_{ijk}, and vice versa. The gradient with respect to β_{ijk} can be found by computing the gradient with respect to θ_{ijk} and then deriving the gradient with respect to β_{ijk} using the chain rule of derivatives. More precisely, the chain rule of derivatives yields

$$\frac{\partial \log P(\mathbf{D} \mid \boldsymbol{\theta})}{\partial \beta_{ijk}} = \sum_{i'j'k'} \frac{\partial \log P(\mathbf{D} \mid \boldsymbol{\theta})}{\partial \theta_{i'j'k'}} \cdot \frac{\partial \theta_{i'j'k'}}{\partial \beta_{ijk}} \,. \tag{10}$$

Since $\partial\theta_{i'j'k'}/\partial\beta_{ijk} = 0$ unless $i = i'$ and $j = j'$, (10) simplifies to

$$\begin{aligned}
\frac{\partial \log P(\mathbf{D} \mid \boldsymbol{\theta})}{\partial \beta_{ijk}} &= \sum_{j'} \frac{\partial \log P(\mathbf{D} \mid \boldsymbol{\theta})}{\partial \theta_{ij'k}} \cdot \frac{\partial \theta_{ij'k}}{\partial \beta_{ijk}} \\
&= \sum_{j'} \frac{\partial \log P(\mathbf{D} \mid \boldsymbol{\theta})}{\partial \theta_{ij'k}} \cdot \frac{\partial \beta_{ij'k}^2/\partial\beta_{ijk} \cdot \sum_l \beta_{ilk}^2 - \left(\partial \sum_l \beta_{ilk}^2\right) / \left(\partial \beta_{ijk}\right) \cdot \beta_{ij'k}^2}{(\sum_l \beta_{ilk}^2)^2}
\end{aligned}$$

$$= \frac{2 \cdot \beta_{ijk}}{(\sum_l \beta_{ilk}^2)^2} \left(\frac{\partial \log P(\mathbf{D} \mid \boldsymbol{\theta})}{\partial \theta_{ijk}} \sum_l \beta_{ilk}^2 - \sum_l \frac{\partial \log P(\mathbf{D} \mid \boldsymbol{\theta})}{\partial \theta_{ilk}} \beta_{ilk}^2 \right) .$$

Now, using the identity $\partial \log P(\mathbf{D})/\partial\theta_{ijk} = ec(\mathbf{fa}_i^{jk})/(\beta_{ijk}^2) \cdot \sum_l \beta_{ilk}^2$ and (8), we get

$$\frac{\partial \log P(\mathbf{D} \mid \boldsymbol{\theta})}{\partial \beta_{ijk}} = 2 \cdot \left(\frac{\mathrm{ec}(\mathbf{fa}_i^{jk} \mid \mathbf{D})}{\beta_{ijk}} - \theta_{ijk} \cdot \sum_l \mathrm{ec}(\mathbf{fa}_i^{lk} \mid \mathbf{D}) \right) . \qquad (11)$$

Again, only local computations are involved. Moreover, many numerical optimization libraries provide advanced gradient methods which need in addition to a procedure to compute the log-likelihood only a procedure to compute the gradient of the log-likelihood function (see e.g. Numerical Recipes source code [25], Matlab [12], and Netlab [21]). Thus, implementing gradient-based maximum likelihood estimators for Bayesian network seems to be only slightly more difficult than implementing the EM algorithm. However, the used gradient-based method should be carefully selected to avoid computational complexity as we will explain in Section 5. Before doing so, let us explain when the EM algorithm is expected to converge slowly.

4 "EM-hard" and "EM-easy" Problems

As stated in the introduction, the EM algorithm converges much faster than simple gradient. In this section, we will briefly review a well-known condition on the data cases when the EM algorithm is expected to slowly converge. For more technical details we refer to [8,17].

Assume that $\langle \boldsymbol{\theta}^k \rangle$ is a sequence of parameters computed by the EM algorithm. Furthermore, assume that $\langle \boldsymbol{\theta}^k \rangle$ converges to some $\boldsymbol{\theta}^*$. Then, in the neighbourhood of $\boldsymbol{\theta}^*$, the EM algorithm is essentially a linear iteration, see e.g. [17]. However, as shown by Dempster et al. [8],

> the greater the proportion of missing information (question mark entries in the data cases, see Section 3), the slower the rate of convergence (in the neighbourhood of $\boldsymbol{\theta}^*$).

Therefore, it is reasonable to classify data cases into "EM-hard" and "EM-easy" problems depending on the fraction of observed and latent parameters of the model. A variables is called *latent*, if the states of the variable are never observed. When this ratio is small, then EM exhibits fast convergence; if the ratio approaches unity, EM will exhibit slow convergence.

5 (Scaled) Conjugate Gradients

As described in Section 3, the simple gradient ascent algorithm uses a fixed step size δ when following the gradient, i.e. in every iteration the parameters are chosen according to

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \delta \cdot \nabla_{\boldsymbol{\theta}} \log L(\mathbf{D}, \boldsymbol{\theta}^k) \ ,$$

where $\boldsymbol{\theta}^k$ denotes the vector of parameters in the k-th iteration. Though it will converge to a (local) maximum for small enough δ, it is not a priori clear how to choose δ. As a consequence, it often converges very slowly. Instead, it would be better to perform a series of *line searches* to choose δ in each iteration, i.e. to do a one dimensional iterative search for δ in the direction of the gradient $\nabla_{\boldsymbol{\theta}^k} := \nabla_{\boldsymbol{\theta}} \log L(\mathbf{D}, \boldsymbol{\theta}^k)$ maximizing $\log L(\mathbf{D}, \boldsymbol{\theta}^k + \delta \cdot \nabla_{\boldsymbol{\theta}^k})$. A common line-search technique is to initially bracket a maximum and then isolate it using an exact algorithm like Brent's Method, but other methods are possible (see e.g. [25]). One of the problems 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.

Conjugate Gradient

Conjugate gradients (CG) compute so-called conjugate directions $h_0, h_1, \ldots$, which are not interfering, and estimate the step size along these directions with line searches. As Nocedal [22] noted, the best variant of conjugate gradients is generally believed to be the *Polak-Ribiere* conjugate gradient. Following the schema of hill climbing, it iteratively performs two steps starting with $\boldsymbol{\theta}_0 \in \mathcal{H}$ and $h_0 = \nabla_{\boldsymbol{\theta}_0}$:

1. (Conjugate directions) Set the next direction h_{k+1} according to $h_{k+1} = \nabla_{\boldsymbol{\theta}^{k+1}} + \gamma_k \cdot h_k$ where
$$\gamma_k = \frac{(\nabla_{\boldsymbol{\theta}^{k+1}} - \nabla_{\boldsymbol{\theta}^k}) \cdot \nabla_{\boldsymbol{\theta}^{k+1}}}{\nabla_{\boldsymbol{\theta}^k} \cdot \nabla_{\boldsymbol{\theta}^k}} \ . \tag{12}$$
2. (Line search) Compute $\boldsymbol{\theta}^{k+1}$ by maximizing $\log L(\mathbf{D}, \boldsymbol{\theta}^k + \delta \cdot h_{k+1})$ in the direction of h_{k+1}.

For a detailed discussion, we refer to e.g. [16,22].

There are still 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 [7]. Thus, the line search dominates the computational costs of conjugate gradients resulting in a serious disadvantage compared to the EM which does one inference per iteration. Therefore, it is not surprising that researchers in the Bayesian network learning community used inexact line search to reduce the complexity [26,23,1] when accelerating EM. However, the use of both exact and inexact line searches within conjugate gradients requires careful considerations [23,22]. Thus, we decided to use a variant of conjugate gradients called *scaled conjugate gradients* due to Møller [18] which led to significant speed up in the context of learning neural networks while preserving accuracy.

Scaled Conjugate Gradients

To underpin the modularity of gradient-based techniques, we will explain scaled conjugate gradients for minimizing a general (scoring) function $f : \mathbb{R}^k \mapsto \mathbb{R}$ as it is implemented e.g. in the Netlab library [21]. Using the log-likelihood as f and substituting f by $-f$ yields the corresponding maximum likelihood parameter estimation for (discrete) Bayesian network. Further details about scaled conjugate gradients can be found in [18,20].

Scaled conjugate gradients (SCGs) substitute the line search 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 per iteration compared to EM. Let $\tilde{f}_{\boldsymbol{\theta}}$ denote the quadratic approximation of f at $\boldsymbol{\theta}$, i.e.

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

Furthermore, let $\boldsymbol{\theta}^k$ be the last parameter estimation, and h_k the current conjugate direction. Then, the step size δ_k minimizing $\tilde{f}_{\boldsymbol{\theta}^k}(\boldsymbol{\theta}^k + \delta \cdot h_k)$ is given by

$$\delta_k = \frac{-h_k^T \cdot \tilde{f}'_{\boldsymbol{\theta}^k}(\boldsymbol{\theta}^k)}{h_k^T \cdot f''(\boldsymbol{\theta}^k) \cdot h_k} \ ,$$

provided that the Hessian $f''(\boldsymbol{\theta}^k)$ is positive definite. Assuming this, the next parameter estimation would be

$$\boldsymbol{\theta}^k = \tilde{f}_{\boldsymbol{\theta}^k}(\boldsymbol{\theta}^k + \delta_k \cdot h_k) \ .$$

However, the computation of the second order term $f''(\boldsymbol{\theta}^k) \cdot h_k$ in (5) is expensive. Therefore, it is approximated by

$$s_k = \frac{f'(\boldsymbol{\theta}^k + \sigma_k \cdot h_k) - f'(\boldsymbol{\theta}^k)}{\sigma_k} \qquad (13)$$

for some small σ_k. The approximation (13) tends in the limit to the true value $f''(\boldsymbol{\theta}^k) \cdot h_k$. As Møller pointed out, the approximation (13) is poor when the current point is far from the desired minimum [9]. The Hessian becomes indefinite, and the search might fail. Møller proposed to introduce a scalar λ_k into (13) to regulate the indefiniteness of $f''(\boldsymbol{\theta}^k)$ by adding $\lambda_k \cdot h_k$ to the right-hand side of (13), i.e.,

$$s_k = \frac{f'(\boldsymbol{\theta}^k + \sigma_k \cdot h_k) - f'(\boldsymbol{\theta}^k)}{\sigma_k} + \lambda_k \cdot h_k \ .$$

Now, there are two cases depending on whether the Hessian is positive definite or not. To test on that, it is enough to evaluate the sign of

$$\alpha_k := h_k^T \cdot s_k \ .$$

In case that $\alpha_k \leq 0$, i.e. the Hessian is indefinite, the value λ_k is raised to ensure positive definiteness. The new value λ_k is set to

$$\bar{\lambda}_k = 2 \cdot \left(\lambda_k - \frac{\alpha_k}{|h_k|^2} \right) ,$$

where $\bar{\lambda}_k$ is the renamed, scaled λ_k. This leads to

$$\bar{\alpha}_k = -\alpha_k + \lambda_k \cdot |h_k|^2 .$$

Otherwise the scaling parameters λ_k and α_k remain unchanged. In both cases, the step size δ_k is given by

$$\delta_k = \frac{\mu_k}{\alpha_k} .$$

where $\mu_k := -h_k^T \cdot f'(\boldsymbol{\theta}^k)$. The value λ_k directly scales the step size δ_k in the way, that the bigger the λ_k, the smaller the step size δ_k.

So far, we have assumed that the (artificially scaled) Hessian is a good approximation. However, even if it is positive definite, the approximation might be poor. To measure the quality of the approximation, Møller considers the relative error

$$\begin{aligned} \Delta_k &= \frac{f(\boldsymbol{\theta}_k) - f(\boldsymbol{\theta}_k + \delta_k \cdot h_k)}{f(\boldsymbol{\theta}_k) - \tilde{f}_{\boldsymbol{\theta}_k}(\delta_k \cdot h_k)} \\ &= \frac{2 \cdot \alpha_k \left(f(\boldsymbol{\theta}_k) - f(\boldsymbol{\theta}_k + \delta_k \cdot h_k)\right)}{-h_k^T \cdot f'(\boldsymbol{\theta}^k)} . \end{aligned}$$

The closer Δ_k is to 1.0, the better is the approximation. Now, the scaling parameter λ_k is raised and lowered according to

$$\lambda_k \leftarrow \begin{cases} 0.25 \cdot \lambda_k & \Delta_k > 0.75 \\ \lambda_k + \frac{(1-\Delta_k)\alpha_k}{|h_k|^2} & \Delta_k < 0.25 \end{cases} .$$

This leads to the overall algorithm as given in Table 1. In general, it may get stuck in local maxima, and may have problems on plateaux and ridges. Well-known techniques such as random restarts or random perturbations are possibilities to avoid them.

6 Experiments

In the experiments described below, we implemented all three algorithms, i.e., EM, conjugate-gradients, and scaled conjugate-gradients using Netica API (`http://www.norsys.com`) for Bayesian network inference. We adapted the conjugate gradient (CG) described in [25] to fullfil the constraints described above. Based on this code, we adapted the scaled conjugate gradient (SCG) as implemented in the Netlab library [21], see also [4]) with an upper bound on the scale parameter of $2 \cdot 10^6$. To see how sensitive the methods to (simple) priors are, we also implemented each of them using BDeu priors [11]. The prior was set to be 1 for all parameters.

Table 1. Basic scaled scaled conjugate gradient algorithm as introduced by Møller [18]. The parameters σ and λ_1 are usually set in between $0 < \sigma \leq 10^{-4}$ and $0 < \lambda_1 \leq 10^{-6}$. The *converged* condition in line 21 loop refers to some stopping criterion such as stopping when a change in log-likelihood is less than some user-specified threshold from one iteration to the next

0: Initialize $\boldsymbol{\theta}_1$, σ, λ_1, and $\bar{\lambda_1} = 0.0$
1: Compute gradient and the conjugate direction h_0 according to (11)
2: $success := true$ and $k = 1$
3: **If** $success = true$ **then** /* *calculate second oder information* */
4: $\sigma_k := \sigma / h_k$
5: $s_k := \left[f'(\boldsymbol{\theta}^k + \sigma_k \cdot h_k) - f'(\boldsymbol{\theta}^k)\right] / \sigma_k + \lambda_k \cdot h_k$
6: $\alpha_k := h_k^T \cdot s_k$
7: Scale α_k, i.e. $\alpha_k := \alpha_k + (\alpha_k - \bar{\alpha}_k)) \cdot |h_k|^2$
8: **if** $\alpha_k \leq 0$ **then** /* *make Hessian positive definite* */
9: $\bar{\lambda}_k := 2 \cdot \left[\lambda_k - \alpha_k / |h_k|^2\right]$
6: $\bar{\alpha}_k := -\alpha_k + \lambda_k \cdot |h_k|^2$
7: $\lambda_k := \bar{\lambda}_k$
8: $\mu_k := -h_k^T \cdot f'(\boldsymbol{\theta}^k)$ /* *Calculate new step size* */
9: $\delta_k := \mu_k / \alpha_k$
10: Measure the quality Δ_k of the approximation
11: **if** $\delta_k \geq 0$ **then** /* *successful reduction in error* */
12: $success := true$
13: Compute new conjugate direction h_k according to (12)
14: $\bar{\lambda}_k := 0$
14: **if** $\Delta_k > 0.75$ **then** /* *reduce scale parameter* */
15: $\lambda_k := 0.25 \cdot \lambda_k$
16: **else** /* *reduction in error is not possible* */
17: $\bar{\lambda}_k := \lambda_k$
18: $success := false$
19: **if** $\Delta_k < 0.25$ **then** /* *increase scale parameter* */
20: $\lambda_k := \lambda_k + (1 - \Delta_k) \cdot \alpha_k / |h_k|^2$
21: **if** converged **terminate**
22: **else** go to step 3

Methodology

Data were generated from four Bayesian networks whose main characteristics are described in Table 2. 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 values missing at random 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. To do so, we initialized each conditional probability table entry as a uniform random sample from $[0, 1]$, and normalized the conditional probability tables afterwards. We ran each algorithm on each data set starting from each of the generated initial sets of parameters. We used a simple, typical stopping

Table 2. Description of the networks used in the experiments

Name	Description
Alarm	Well-known benchmark network for the ICU ventilator management. It consists of 37 nodes and 752 parameters [2]. 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 [3].
3-1-3, **5-3-5**	Two artificial networks with a feed-forward architecture known from neural networks [3]. 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.

criterion. We stopped when a limit of 200 iterations 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 to see the behaviour of the algorithms when they are close to a solution. The EM algorithm can be very slow if high accuracy of the estimate is necessary, whereas (advanced) 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 [5].

All learned models were evaluated on the test set using a standard measure, the *normalized-loss*

$$\frac{1}{N}\sum_{l=1}^{N}(\log P^*(\mathbf{d}_l) - \log P(\mathbf{d}_l)) ,$$

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 running time. The latter is dominated by the Bayesian network inference, because except for the likelihood evaluations all computations are linear. Therefore, we consider *EM-equivalent* iterations, i.e. likelihood evaluations. Each iteration of the scaled conjugate gradient consists of two EM-equivalent iterations.

Results

The results on the normalized-loss measure are shown in Tables 3 and 4. Without setting priors, SCG clearly outperformed EM, as Table 3 shows. Therefore, we concentrate in the evaluation on the results with priors, cf. Table 4. We omit CG from further investigations because it did not terminated in reasonable time on **Insurance** and **Alarm**. No run was finished after one week on a Pentium III, 450 MHz. However, on **3-1-3** and **5-3-5**,

Table 3. Average normalized-loss of **Insurance**, **Alarm**, **3-1-3**, and **5-3-5** on an independent test set consisting of 10000 data cases **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

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

CG showed the expected behaviour. Considering only the normalized-loss, it reached normalized-losses slightly closer to zero than EM and SCG (mean difference around 0.001). The number of iterations on **3-1-3** was on average two times higher than for EM, on **5-3-5** they were on average twice lower. However, the number of EM-equivalent iterations was in all cases much higher than for EM and SCG. Furthermore, CG was sensitive to the initial set of parameters.

Table 4. Average normalized-loss of **Insurance**, **Alarm**, **3-1-3**, and **5-3-5** on an independent test set consisting of 10000 data cases **with BDeu priors**. The mean and standard deviation of five restarts are reported. 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

Domain / # hidden (in %)	Meth.	Number of data cases			
		100	200	500	1000
Insurance	EM	0.99 ± 0.04	**0.64 ± 0.03**	0.37 ± 0.02	0.20 ± 0.01
0.0	SCG	**1.82 ± 1.76**	0.62 ± 0.02	**0.33 ± 0.03**	**0.51 ± 0.60**
0.1	EM	1.08 ± 0.05	0.68 ± 0.04	0.37 ± 0.02	**0.23 ± 0.01**
	SCG	**1.14 ± 0.35**	**0.69 ± 0.11**	**0.89 ± 0.67**	0.30 ± 0.08
0.2	EM	**1.07 ± 0.03**	0.76 ± 0.03	0.45 ± 0.02	0.25 ± 0.02
	SCG	1.35 ± 0.48	**0.76 ± 0.06**	**0.44 ± 0.06**	**0.37 ± 0.24**
0.3	EM	1.35 ± 0.04	0.94 ± 0.02	**0.52 ± 0.03**	0.29 ± 0.01
	SCG	**1.59 ± 0.71**	**1.25 ± 0.39**	0.62 ± 0.13	**0.38 ± 0.09**
Alarm	EM	0.81 ± 0.00	**0.62 ± 0.00**	**0.29 ± 0.00**	0.16 ± 0.00
0.0	SCG	**1.01 ± 0.25**	0.94 ± 0.37	0.33 ± 0.02	**0.17 ± 0.00**
0.1	EM	1.28 ± 0.00	0.68 ± 0.00	0.31 ± 0.00	0.18 ± 0.00
	SCG	**1.81 ± 0.74**	**0.87 ± 0.27**	**0.31 ± 0.00**	**0.19 ± 0.00**
0.2	EM	**1.10 ± 0.00**	0.64 ± 0.00	0.31 ± 0.00	**0.24 ± 0.00**
	SCG	1.31 ± 0.19	**1.25 ± 0.80**	**0.31 ± 0.00**	0.24 ± 0.01
0.3	EM	1.22 ± 0.00	**0.65 ± 0.00**	**0.41 ± 0.00**	**0.21 ± 0.00**
	SCG	**1.28 ± 0.08**	0.66 ± 0.01	1.12 ± 0.96	0.22 ± 0.00
3-1-3	EM	**0.15 ± 0.01**	0.07 ± 0.00	**0.03 ± 0.00**	0.02 ± 0.00
0.0	SCG	0.16 ± 0.02	**0.07 ± 0.00**	0.03 ± 0.00	**0.02 ± 0.00**
	CG	0.17 ± 0.04	0.07 ± 0.00	0.04 ± 0.02	0.02 ± 0.00
0.1	EM	0.13 ± 0.02	0.13 ± 0.03	0.05 ± 0.00	**0.02 ± 0.00**
	SCG	**0.13 ± 0.01**	**0.11 ± 0.01**	**0.06 ± 0.03**	0.02 ± 0.00
	CG	0.13 ± 0.02	0.11 ± 0.02	0.05 ± 0.00	0.02 ± 0.01
0.2	EM	**0.20 ± 0.01**	**0.16 ± 0.04**	**0.05 ± 0.00**	**0.03 ± 0.00**
	SCG	0.25 ± 0.05	0.10 ± 0.01	0.05 ± 0.00	0.04 ± 0.03
	CG	0.20 ± 0.03	0.10 ± 0.01	0.06 ± 0.00	0.03 ± 0.00
0.3	EM	**0.15 ± 0.00**	**0.16 ± 0.01**	**0.03 ± 0.00**	0.05 ± 0.05
	SCG	0.17 ± 0.02	0.17 ± 0.01	0.04 ± 0.03	**0.02 ± 0.00**
	CG	0.16 ± 0.02	0.18 ± 0.04	0.03 ± 0.00	0.02 ± 0.00
5-3-5	EM	0.23 ± 0.01	0.22 ± 0.01	0.15 ± 0.01	0.11 ± 0.00
0.0	SCG	**0.19 ± 0.03**	**0.21 ± 0.01**	**0.13 ± 0.02**	**0.10 ± 0.01**
	CG	0.14 ± 0.07	0.13 ± 0.07	0.09 ± 0.05	0.06 ± 0.03
0.1	EM	0.19 ± 0.01	0.21 ± 0.01	0.18 ± 0.00	0.13 ± 0.01
	SCG	**0.14 ± 0.04**	**0.20 ± 0.01**	**0.15 ± 0.02**	**0.09 ± 0.02**
	CG	0.12 ± 0.06	0.13 ± 0.07	0.11 ± 0.06	0.08 ± 0.04
0.2	EM	0.16 ± 0.01	0.18 ± 0.01	0.16 ± 0.01	0.14 ± 0.01
	SCG	**0.12 ± 0.03**	**0.14 ± 0.03**	**0.14 ± 0.02**	**0.13 ± 0.01**
	CG	0.09 ± 0.05	0.10 ± 0.05	0.09 ± 0.05	0.08 ± 0.04
0.3	EM	0.25 ± 0.01	0.17 ± 0.00	0.25 ± 0.01	0.15 ± 0.00
	SCG	**0.20 ± 0.04**	**0.16 ± 0.01**	**0.21 ± 0.03**	**0.13 ± 0.02**
	CG	0.15 ± 0.08	0.10 ± 0.05	0.15 ± 0.08	0.09 ± 0.05

Quality: EM and SCG reached similar normalized-losses. As expected, a higher number of data cases led to a lower normalized-loss. Similarly, priors led to a lower normalized-loss. The absolute differences in averaged normalized-losses between using priors and not using priors are much smaller for SCG than for EM. Both were relatively insensitive to the initial set of parameters. The EM tended to have a slightly lower variation, but a *one-tailed, paired sampled t test* over all experiments (with priors) showed that

Table 5. Average EM equivalent iterations performed by EM and SCG. 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

Domain / # hidden (in %)	Method	Number of data cases: 100	200	500	1000
Insurance 0.0	EM	$100 \pm 28(144 \pm 34)$	$143 \pm 40(199 \pm 0)$	$188 \pm 17(199 \pm 0)$	$188 \pm 13(199 \pm 0)$
	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 0.0	EM	$1 \pm 0(1 \pm 0)$	$1 \pm 0(1 \pm 0)$	$1 \pm 0(1 \pm 0)$	$1 \pm 0(1 \pm 0)$
	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 0.0	EM	$21 \pm 10(27 \pm 7)$	$21 \pm 2(22 \pm 2)$	$18 \pm 3(22 \pm 4)$	$16 \pm 1(18 \pm 1)$
	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 0.0	EM	$97 \pm 7(38 \pm 6)$	$92 \pm 40(52 \pm 13)$	$120 \pm 39(78 \pm 17)$	$113 \pm 24(85 \pm 7)$
	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	$72 \pm 42(170 \pm 21)$	$88 \pm 20(179 \pm 29)$	$121 \pm 28(151 \pm 9)$	$130 \pm 26(143 \pm 22)$

only a 0.045 difference in mean of EM and SCG ($p = 0.05$) 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.* The sign test yielded that the number of cases, where the best model of SCG outperformed EM's best model, was large, 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.

Number of Iterations: Table 5 summarizes the number of EM-equivalent iterations. EM was significantly faster than SCG on **Alarm** . However, this did not carry over to all domains. A *one-tailed, paired sampled t test* (over all corresponding experiments with priors) 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**.

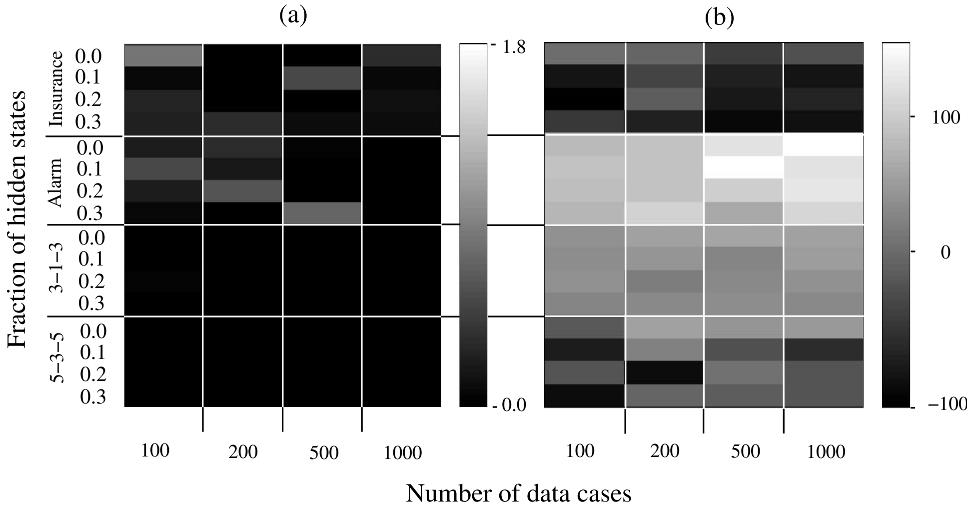


Fig. 2. Differences between SCG's and EM's behaviours. In both pictures, the mean of five restarts were subtracted. (**a**) Differences (SCG − EM) of average normalized-loss over **Insurance**, **Alarm**, **3-1-3**, and **5-3-5** on an independent test set consisting of 10000 data cases. (**b**) Difference (SCG − EM) of EM equivalent iterations

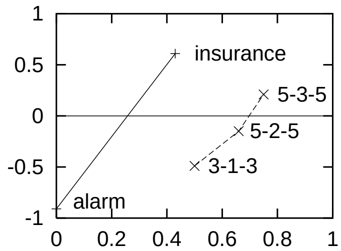
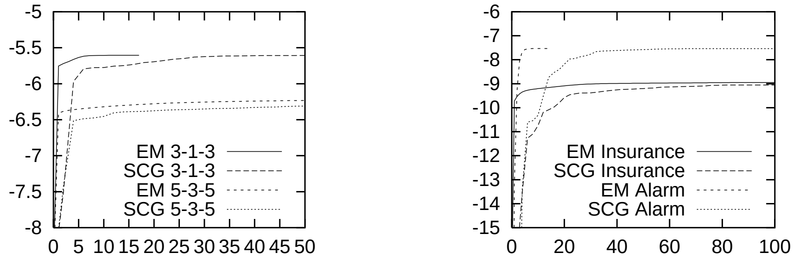


Fig. 3. Percentage of latent parameters vs. SCG's speed-up. The domains are grouped into network structures with (solid line) and without (dashed line) encoded prior information. Both curves confirm the theory that the more latent parameters in the data cases, the slower the convergence of the EM algorithm (when compared to scaled conjugate gradients)

Thus, an acceleration of conjugate gradients is possible. Next to the quality of the estimated parameters, SCG seems to have some EM-like characteristics. It is fast, and (relatively) insensitive to the initial set of parameters. This is remarkable given that gradients are usually considered to perform worse than EM (on discrete Bayesian networks). However, the behaviour of both SCG and EM varied over the domain.



(a) (b)

Fig. 4. Averaged learning curves. (**a**) Learning curves averaged over five restarts for 1000 data cases, no missing values. (**b**) Learning curves averaged over five restarts for 1000 data cases. Fraction of 0.3 missing values for **Alarm**, no missing values for **Insurance**

"EM-easy" vs. "EM-hard": As explained in Section 4, it is reasonable to classify domains into "EM-easy" and "EM-hard" ones. Figure 2 shows the differences in average normalized-loss and in EM equivalent iterations. **Alarm** and **Insurance** are networks where prior knowledge is encoded into the structure, whereas **3-1-3** and **5-3-5** have uninformative structures. Figure 2 (a) shows that on average EM tended to yield better normalized-loss on prior knowledge networks. However, within both types of network structures, the fraction of latent parameters seems to correlate to "EM-hard" domains (in particular with respect to the number of iterations). This was predicted by the theory, cf. Section 4. Figure 2 (b) shows that SCG converged faster on networks with a higher number of latent parameters, i.e. **Insurance** and **5-3-5**. Figure 3(a) shows the percentage of latent parameters vs. speed-up which is defined as the quotient of average EM-equivalent iterations (over corresponding restarts) of EM and SCG minus one. To test whether SCGs performs well on "EM-hard" domains, i.e. that these domains are "SCG-easy" ones, 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. SCG's 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, Figs. 4(b) and 4(c) show that the EM algorithm was 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 SCG is not only competitive but that it can be superior to EM, and in contrast to CG, it learned **Insurance** and **Alarm**.

7 Related Work

Both, EM and gradients are well-known parameter estimation techniques. For a general introduction see [17,16]. Bishop [4] and Møller [18] show how to use (scaled) conjugate gradients to estimate parameters of neural networks. Lauritzen [15] introduced EM for Bayesian networks (see also Heckerman [10] for a nice tutorial). The original work on gradient-based approaches in the context of Bayesian networks was done by Binder *et al.* [3], some recent work by Jensen [13,14]. However, we are neither aware of any direct experimental comparison between the two approaches nor of research investigating accelerated conjugate gradients such as scaled ones. Binder *et al.* [3] used traditional conjugate gradients, and did not compare more advanced gradient-based and EM-based approaches. Bauer *et al.* [1] 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. Jensen [13] derived a formula for the gradient which relates it to sensitivity analyses of Bayesian networks. Jensen did not compare it with the EM. Thiesson [26] discussed (traditional) conjugate gradient accelerations of the EM, but does not report on experiments. Ortiz and Kaelbling [23] conducted experiments with several learning algorithms (mainly accelerated EMs) for 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 classified 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 gradients.

8 Conclusions

In this case study, we experimentally compared EM and gradient-based algorithms for maximum likelihood parameter estimation on several Bayesian networks. To overcome the expensive line search of traditional conjugate gradients, we applied *scaled conjugate gradients*. They are novel in the context of Bayesian networks.

The experiments show that scaled conjugate gradients are competitive with EM in both quality of the estimate and convergence speed. To the best of our knowledge, this is the first time that gradient-based techniques are reported to be competitive with EM with respect to quality and speed for discrete Bayesian networks. 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 (in theory and in practice) "EM-easy" and "EM-hard" domains. Scaled conjugate gradients seem especially well suited in the presence of many latent parameters, i.e. “EM-hard” instance. Given that gradients are more flexible – as they allow us to learn non-multinomial parameterization (e.g. noisy-or) using the chain-rule of derivatives and conditional probability distributions using Bayes’ rule – and that they are strongly related to methods in neural networks training, scaled conjugate gradients are a promising alternative to EM for parameter estimation of (discrete) Bayesian networks. Our results confirm that the number of latent parameters determines “EM-hard” instances.

A very promising line of future research seems to be accelerating EM using scaled conjugate gradients, and applying stochastic scaled conjugate gradients [19] to the problem of *online-learning* of Bayesian networks.

Acknowledgements The authors would like to thank Luc De Raedt and Wolfram Burgard for helpful discussions. The first author was supported by the European Union IST programme, IST-2001-33053 (Application of Probabilistic Inductive Logic Programming - APRIL).

References

1. E. Bauer, D. Koller, and Y. Singer. Update Rules for Parameter Estimation in Bayesian Networks. In D. Geiger and P. P. Shenoy, editors, *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 3–13, Providence, Rhode Island, USA, 1997. Morgan Kaufmann.
2. I. Beinlich, H. Suermondt, R. Chavez, and G. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In J. Hunter, editor, *Proceedings of the Second European Conference on Artificial Intelligence and Medicine (AIME-89)*, volume 38 of *Lecture Notes in Medical Informatics*, pages 247–256, City University, London, UK, 1989. Springer-Verlag.
3. J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive Probabilistic Networks with Hidden Variables. *Machine Learning*, 29(2/3):213–244, 1997.
4. C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.
5. H. Chan and A. Darwiche. When do numbers really matter? *Journal of Artificial Intelligence Research (JAIR)*, 17:265–287, 2002.
6. D. M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks is NP-hard. Technical Report MSR-TR-94-17, Microsoft Research, Advanced Technology Division, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, USA, 1994.
7. G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.
8. 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.
9. P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Acadamic Press, 1981.

10. D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Advanced Technology Division, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, USA, 1995.
11. 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.
12. Mathworks Inc. Optimization Toolbox 2.2 for Matlab. `http://www.mathworks.com`.
13. F. V. Jensen. Gradient descent training of bayesian networks. In A. Hunter and S. Parsons, editors, *Proceedings of the Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU-99)*, volume 1638 of *Lecture Notes in Computer Science*, pages 190–200. Springer, 1999.
14. F. V. Jensen. *Bayesian networks and decision graphs.* Springer-Verlag, 2001.
15. S. L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.
16. D. G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, 2. edition, 1984.
17. G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions.* Wiley, New York, 1997.
18. M. Møller. A Scaled Conjugate Gradient Algoritm for Fast Supervised Learning. *Neural Networks*, 6:525–533, 1993.
19. M. Møller. Supervised Learning on Large Redundant Training sets. *International Journal Neural Systems*, 4(1):15–25, 1993.
20. M. Møller. *Efficient Training of Feed-Forward Neural Networks.* PhD thesis, Computer Science Department, Aarhus University, Denmark, 1997.
21. I. Nabney. *NETLAB: Algorithms for Pattern Recognition.* Advances in Pattern Recognition. Springer-Verlag, 2001. `http://www.ncrg.aston.ac.uk/netlab/`.
22. J. Nocedal. Theory of algorithms for unconstrained optimization. *Acta Numerica*, pages 199–242, 1992.
23. L. E. Ortiz and L. P. Kaelbling. Accelerating EM: An Empirical Study. In K. B. Laskey and H. Prade, editors, *Proceedings of the Fifteenth Annual Conference on Uncertainty in Articial Intelligence (UAI-99)*, pages 512–521, Stockholm, Sweden, 1999. Morgan Kaufmann.
24. J. Pearl. *Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 2. edition, 1991.
25. 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. `http://www.nr.com`.
26. B. Thiesson. Accelerated quantification of Bayesian networks with incomplete data. In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings of First International Conference on Knowledge Discovery and Data Mining*, pages 306–311, Montreol, Canada, 1995. AAAI Press.