Scaled CGEM: A Fast Accelerated EM

Jörg Fischer and Kristian Kersting

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

Abstract. The EM algorithm is a popular method for maximum likelihood estimation of Bayesian networks in the presence of missing data. Its simplicity and general convergence properties make it very attractive. However, it sometimes converges slowly. Several accelerated EM methods based on gradient-based optimization techniques have been proposed. In principle, they all employ a line search involving several NP-hard likelihood evaluations. We propose a novel acceleration called SCGEM based on scaled conjugate gradients (SCGs) well-known from learning neural networks. SCGEM avoids the line search by adopting the scaling mechanism of SCGs applied to the expected information matrix. This guarantees a single likelihood evaluation per iteration. We empirically compare SCGEM with EM and conventional conjugate gradient accelerated EM. The experiments show that SCGEM can significantly accelerate both of them and is equal in quality.

1 Introduction

Bayesian networks [19] are one of the most important frameworks for representing and reasoning with probabilities. They specify joint probability distributions over finite sets of random variables, and have been applied to many real-world problems in diagnosis, forecasting, sensor fusion etc. Over the past years, there has been much interest in the problem of learning Bayesian networks from data. 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 [6].

It is often desired to find the parameters *maximizing the likelihood* (ML). The likelihood 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 maximum likelihood estimate typically cannot be written in closed form. It is a numerical optimization problem, and all known algorithms involve nonlinear, iterative optimization and multiple calls to a Bayesian network inference as subroutines. The latter ones have been proven to be NP-hard [4]. The most common technique for ML parameter estimation of Bayesian networks in the presence of missing data is the Expectation-Maximization (EM) algorithm.

Despite the success of the EM algorithm in practice due to its simplicity and fast initial progress, it has been argued (see e.g. [8, 13] and references in there) that the EM convergence can be extremely slow, and that more advanced second-order methods should in general be favored to EM. In the context of Bayesian networks, Thiesson [21], Bauer et al. [1], and Ortiz and Kaelbling [17] investigated acceleration of the EM algorithm. All approaches rely on conventional (gradient-based) optimization techniques viewing the change in values in the parameters at an EM iteration as generalized gradient (see [18] for a nice overview). Gradient ascent yields *parameterized EM*, and conjugate gradient yields *conjugate gradient EM* (CGEM). Although the accelerated EMs can significantly speed-up the EM, they all require more computational efforts than the basic EM. One reason is that they perform in each iteration a line search to choose an optimal step size. There are drawbacks of doing a line search. First, a line search introduces new problem-dependent parameters such as stopping criterion. Second, the line search involves several likelihood evaluations which are NP-hard for Bayesian networks. Thus, the line search dominates the computational costs resulting in a disadvantage of the accelerated EMs compared to the EM which does one likelihood evaluation per iteration. The computational extra costs have to be amortized over the long run to gain a speed-up.

The contribution of the present paper is a novel acceleration of EM called *scaled CGEM* (SCGEM) which overcomes the expensive line search. It evaluates the likelihood as often as the EM per iteration namely once. This also explains the title of the paper "A Fast Accelerated EM". SCGEM adopt the ideas underlying *scaled conjugate gradients* (SCGs) which are well-known from the field of learning neural networks [15]. SCGs employ an approximation of the Hessian of the scoring function to quadratically extrapolate the minimum instead of doing a line search. Then, a Levenberg-Marquardt approach [12] scales the step size. SCGEM adopts the scaling mechanism for maximization and applies it to the expected information matrix. This type of accelerated CGEM is novel. Other work for learning Bayesian networks investigated only approximated line searches [21, 17, 1], thus evaluates the likelihood at least twice per iteration. From the experimental results, we will argue that SCGEM can significantly accelerate both EM and CGEM, and is equal in quality.

We proceed as follows. After briefly introducing Bayesian networks in Section 2, we review maximum likelihood estimation via the EM and gradient ascent in Section 3. In Section 4, we motivate why accelerating EM is important and review basic acceleration techniques. Afterwards we introduce SCGEM in Section 5. In Section 6, we experimentally compare the SCGEM algorithm with the EM and CGEM algorithms. Before concluding, we discuss related work.

2 Bayesian Networks

Throughout the paper, we will use X to denote a random variable, x to denote a state and $\mathbf{X}$ (resp. $\mathbf{x}$) to denote 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* [19] 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 $x_i^1, \ldots, x_i^{j_i}$ of possible states. 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. We denote the parents of X_i in a graph-theoretical sense by $\mathbf{Pa}_i$. The family of X_i is $\mathbf{Fa}_i := \{X_i\} \cup \mathbf{Pa}_i$. With each node X_i, a conditional probability table is associated specifying the distribution $\mathbf{P}(X_i \mid \mathbf{Pa}_i)$. The table entries are $\theta_{ijk} = P(X_i = x_i^j \mid \mathbf{Pa}_i = \mathbf{pa}_i^k)$, where $\mathbf{pa}_i^k$ denotes the kth joint state of the X_i's parents. 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 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 with given structure by the M-dimensional vector $\boldsymbol{\theta}$ consisting of all θ_{ijk}'s.

3 Basic ML Parameter 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 $\boldsymbol{\theta}$. 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. We assume that the data cases are independently sampled from identical distributions (iid. We seek those parameters $\boldsymbol{\theta}^*$ which maximize the likelihood $L(\mathbf{D}, \boldsymbol{\theta}) := P(\mathbf{D} \mid \boldsymbol{\theta})$ of the data. Due to the monotonicity of the logarithm, we can also seek the parameters maximizing the log-likelihood $LL(\mathbf{D}, \boldsymbol{\theta}) := \log P(\mathbf{D} \mid \boldsymbol{\theta})$. This simplifies because of the iid assumption to $\boldsymbol{\theta}^* = \mathrm{argmax}_{\boldsymbol{\theta} \in \mathcal{H}} \sum_{l=1}^{N} \log P(\mathbf{d}_l \mid \boldsymbol{\theta})$. Thus, the search space $\mathcal{H}$ to be explored is spanned by the space over the possible values of $\boldsymbol{\theta}$. In case of complete data $\mathbf{D}$, i.e., the values of all random variables are observed, Lauritzen [11] showed that maximum likelihood estimation simply corresponds to frequency counting. However, in the presence of missing data the maximum likelihood estimates typically cannot be written in closed form, and iterative optimization schemes like the EM or gradient-based algorithms are needed. We will now briefly review both approaches because SCGEM heavily builds on them.

The *EM algorithm* [5] is a classical approach to maximum likelihood estimation in the presence of missing values. The basic observation underlying the Expectation-Maximization algorithm is: learning would be easy if we knew the values for all the random variables. Therefore, it iteratively performs two steps to find the maximum likelihood parameters of a model: **(E-Step)** 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. **(M-Step)** Each completion is then treated as a fully-observed data case weighted by its probability. New parameters are then computed based on frequency counts.

More formally, the E-step consists of computing the expectation of the likelihood given the old parameters $\boldsymbol{\theta}^n$ and the observed data $\mathbf{D}$, i.e.,

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^n, \mathbf{D}) = E\left[\log P(\mathbf{Z} \mid \boldsymbol{\theta}) \mid \boldsymbol{\theta}^n, \mathbf{D}\right] \ . \tag{1}$$

Here, $\mathbf{Z}$ is a random vector denoting the completion of the data cases $\mathbf{D}$. The current parameters $\boldsymbol{\theta}^n$ and the observed data $\mathbf{D}$ give us the conditional distribution governing the unobserved states. The expression $E[\cdot|\cdot]$ denotes the expectation over this conditional distribution. Q is sometimes called the *expected information matrix*. In the M-step, $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^n, \mathbf{D})$ is maximized w.r.t. $\boldsymbol{\theta}$, i.e., $\boldsymbol{\theta}^{n+1} = \text{argmax}_{\boldsymbol{\theta}}\, Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^n, \mathbf{D})$. Lauritzen [11] showed that this leads to $\theta^*_{ijk} = \text{ec}(\mathbf{fa}_i^{jk} \mid \mathbf{D}) / \text{ec}(\mathbf{pa}_i^j \mid \mathbf{D})$ where $\mathbf{fa}_i^{jk}$ is the joint state consisting of the jth state of variable X_i and the kth joint state $\mathbf{pa}_i^k$ of $\mathbf{Pa}_i$. The term $\text{ec}(\mathbf{a} \mid \mathbf{D})$ denotes the *expected counts* of the joint state $\mathbf{a}$ given the data. They are computed by $\text{ec}(\mathbf{a} \mid \mathbf{D}) = \sum_{l=1}^N P(\mathbf{a} \mid \mathbf{d}_l, \boldsymbol{\theta}^n)$ where any Bayesian network inference engine can be used to compute $P(\mathbf{a} \mid \mathbf{d}_l, \boldsymbol{\theta}^n)$.

Gradient ascent , also known as *hill climbing*, is a classical method for finding a maximum of a (scoring) function. It iteratively performs two steps. First, it computes the *gradient* vector $\nabla_{\boldsymbol{\theta}}$ of partial derivatives of the log-likelihood with respect to the parameters of a Bayesian network at a given point $\boldsymbol{\theta} \in \mathcal{H}$. 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. [12]. Thus, we have to compute the partial derivatives of $LL(\mathbf{D}, \boldsymbol{\theta})$ with respect to θ_{ijk} . According to Binder *et al.* [3], they are given by

$$\frac{\partial LL(\mathbf{D}, \boldsymbol{\theta})}{\partial \theta_{ijk}} = \sum\nolimits_{l=1}^{N} \frac{P(\mathbf{fa}_i^{jk} \mid \mathbf{d}_l, \boldsymbol{\theta})}{\theta_{ijk}} = \frac{\text{ec}(\mathbf{fa}_i^{jk} \mid \mathbf{D})}{\theta_{ijk}} \ . \tag{2}$$

In contrast to EM, 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 $\sum_j \theta_{ijk} = 1$. A general solution is to reparameterize the problem so that the new parameters automatically respect the constraints on θ_{ijk} no matter what their values are. More precisely, we define the parameters $\beta_{ijk} \in \mathbb{R}$ such that $\theta_{ijk} = exp(\beta_{ijk}) / (\sum_l exp(\beta_{ilk}))$ where the β_{ijk} are indexed like θ_{ijk}. It can be shown using the chain rule of derivatives that

$$\begin{aligned}\frac{\partial LL(\mathbf{D}, \boldsymbol{\theta})}{\partial \beta_{ijk}} &= \sum\nolimits_{i'j'k'} \frac{\partial LL(\mathbf{D}, \boldsymbol{\theta})}{\partial \theta_{i'j'k'}} \cdot \frac{\partial \theta_{i'j'k'}}{\partial \beta_{ijk}} = \sum\nolimits_{i'j'k'} \frac{\text{ec}(\mathbf{fa}_{i'}^{j'k'} \mid \mathbf{D})}{\theta_{i'j'k'}} \cdot \frac{\partial \theta_{i'j'k'}}{\partial \beta_{ijk}} \\ &= \text{ec}(\mathbf{fa}_i^{jk} \mid \mathbf{D}) - \theta_{ijk} \sum\nolimits_l \text{ec}(\mathbf{fa}_i^{lk} \mid \mathbf{D}) \ .\end{aligned}$$

An important view of the gradient for our purposes is the following. It highlights the close connection between gradient ascent and the EM:

$$\frac{\partial Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}', \mathbf{D})}{\partial \beta_{ijk}} = \frac{\partial}{\partial \beta_{ijk}} E\left[\sum\nolimits_{i'j'k'} \text{c}(\mathbf{fa}_{i'}^{j'k'} \mid \mathbf{D}) \cdot \log \boldsymbol{\theta}_{i'j'k'} \;\middle|\; \boldsymbol{\theta}', \mathbf{D}\right]$$

$$= \frac{\partial}{\partial \beta_{ijk}} \sum\nolimits_{i'j'k'} \mathrm{ec}(\mathbf{fa}_{i'}^{j'k'} \mid \mathbf{D}) \cdot \log \boldsymbol{\theta}_{i'j'k'} = \sum\nolimits_{i'j'k'} \mathrm{ec}(\mathbf{fa}_{i'}^{j'k'} \mid \mathbf{D}) \frac{\partial \log \boldsymbol{\theta}_{i'j'k'}}{\partial \beta_{ijk}}$$

$$= \sum\nolimits_{i'j'k'} \frac{\mathrm{ec}(\mathbf{fa}_{i'}^{j'k'} \mid \mathbf{D})}{\theta_{i'j'k'}} \cdot \frac{\partial \theta_{i'j'k'}}{\partial \beta_{ijk}} = \frac{\partial \log P(\mathbf{D} \mid \boldsymbol{\theta})}{\partial \beta_{ijk}} \, . \tag{3}$$

Thus, the gradient of the likelihood coincides with the gradient of the expected information matrix [1].

4 Accelerated ML Parameter Estimation

It has been argued that the EM convergence can be extremely slow, see e.g. [13, 8] and references in there. Assume that $\langle \boldsymbol{\theta}^n \rangle$ is a sequence of parameters computed by the EM algorithm. Furthermore, assume that $\langle \boldsymbol{\theta}^n \rangle$ converges to some $\boldsymbol{\theta}^*$. Then, in the neighbourhood of $\boldsymbol{\theta}^*$, the EM algorithm is essentially a linear iteration. However, as shown by Dempster et al. [5], the greater the proportion of missing information, the slower the rate of convergence (in the neighbourhood of $\boldsymbol{\theta}^*$). If the ratio approaches unity, EM will exhibit slow convergence. Therefore, more advanced second-order methods should in general be favored to EM.

In the context of Bayesian networks, Kersting and Landwehr [9] empirically compared second-order gradient techniques with the EM. They were able to show that the former ones can be competitive with EM, but EM remaines the domain independent method of choice for ML estimation because of its simplicity and fast initial progress. Within the Bayesian network learning community several accelerated EM algorithms have been proposed and empirically compared with the EM algorithm [21, 17, 1]. We will now briefly review the basic acceleration techniques following [17] which are also the basic ingredients of SCGEM.

The most straightforward way to accelerate the EM is the *parameterized EM* (PEM). It is a gradient ascent where instead of following the gradient of the log-likelihood we follow the generalized [2] gradient $g_n := \boldsymbol{\theta}_{\mathrm{EM}}^{n+1} - \boldsymbol{\theta}^n$, where $\boldsymbol{\theta}_{\mathrm{EM}}^{n+1}$ denotes the EM update with respect to the parameters after the $n+1$-th iteration. The simple PEM uses a fixed step size δ when following the gradient, i.e., in every iteration the parameters are chosen according to $\boldsymbol{\theta}^{n+1} = \boldsymbol{\theta}^n + \delta \cdot g_n$. 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 in each iteration, i.e., to do a one dimensional iterative search for δ in the direction of the generalized gradient maximizing $LL(\mathbf{D}, \boldsymbol{\theta}^n + \delta \cdot g_n)$. One of the problems with the resulting algorithm is that a maximization in one direction could spoil past maximizations. This problem is solved by *conjugate gradient* EM.

Conjugate gradient EM (CGEM) computes so-called conjugate directions $h_0, h_1, \ldots$, which are orthogonal, and estimate the step size along these directions with line searches [7]. Following the scheme of PEM, it iteratively performs two steps starting with $\boldsymbol{\theta}^0 \in \mathcal{H}$ and $h_0 = g_0$:

---

[1] This means that gradient ascent is a so-called generalized EM.

[2] Generalized gradients perform regular gradient techniques in a transformed space.

1. (Conjugate directions) Set the next direction h_{n+1} according to $h_{n+1} = g_{\boldsymbol{\theta}^{n+1}} - \gamma_n \cdot h_n$ where

$$\gamma_n = \frac{(\nabla LL(\mathbf{D}, \boldsymbol{\theta}^{n+1})^T - \nabla LL(\mathbf{D}, \boldsymbol{\theta}^n)^T) \cdot g_{\boldsymbol{\theta}^{n+1}}}{(\nabla LL(\mathbf{D}, \boldsymbol{\theta}^{n+1})^T - \nabla LL(\mathbf{D}, \boldsymbol{\theta}^n)^T) \cdot h_n} . \tag{4}$$

2. (Line search) Compute $\boldsymbol{\theta}^{n+1}$ by maximizing $LL(\mathbf{D}, \boldsymbol{\theta}^n + \delta \cdot h_{n+1})$ in the direction of h_{k+1}.

Usually, some initial EM steps are taken before switching to the CGEM. We refer to [7, 13] for more details on CGEM.

There are still drawbacks of doing a line search. First, it introduces new problem-dependent parameters such as a stopping criterion. Second, the line search involves several likelihood evaluations, i.e., network inferences which are known to be NP-hard [4]. Thus, the line search dominates the computational costs of CGEM resulting in a serious disadvantage compared to the EM which does one likelihood estimation per iteration. Therefore, it is not surprising that researchers in the Bayesian network learning community used inexact line search to reduce the complexity [21, 17, 1] when accelerating EM. Nevertheless, they require at least one additional likelihood evaluation per iteration compared to the EM. In the next section, we will show how to avoid the line search. We will adopt a variant of conjugate gradients called *scaled conjugate gradients* to accelerate EM. They are due to Møller [15] and led to significant speed up of learning neural networks while preserving accuracy.

5 Scaled Conjugate Gradient EM

Scaled conjugate gradient (SCG) substitutes the line search by employing an approximation of the Hessian of the error function to quadratically extrapolate the minimum using a Levenberg-Marquardt approach [12] to scale the step size. We will now apply this idea to CGEM in the context of maximum likelihood parameter estimation of Bayesian networks. Due to space restrictions, we will discuss the basic idea only. For more details about SCG, we refer to [15].

We quadratically approximate $LL(\mathbf{D}, \boldsymbol{\theta}^n)$ at $\mathbf{x}$, i.e., $LL(\mathbf{D}, \boldsymbol{\theta}^n) + \nabla LL(\mathbf{D}, \boldsymbol{\theta}^n)^T \cdot \mathbf{x} + \frac{1}{2}\mathbf{x}^T \cdot \nabla^2 LL(\mathbf{D}, \boldsymbol{\theta}^n) \cdot \mathbf{x}$. Then, the step size for CGEM is $(h_n^T \cdot \nabla LL(\mathbf{D}, \boldsymbol{\theta}^n))/(h_n^T \cdot \nabla^2 LL(\mathbf{D}, \boldsymbol{\theta}^n) \cdot h_n)$. However, one has to compute and store the Hessian matrix $\nabla^2 LL(\mathbf{D}, \boldsymbol{\theta}^n)$. To avoid this, Møller estimated $\nabla^2 LL(\mathbf{D}, \boldsymbol{\theta}^n) \cdot h_n$ with a Newton quotient. The approximation needs to be negative definite in order to obtain a maximum. Therefore, a scalar λ_n is introduced to assure the definiteness, i.e.,

$$s_n := \frac{\nabla LL(\mathbf{D}, \boldsymbol{\theta}^n + \sigma^n \cdot h_{k+1}) - \nabla LL(\mathbf{D}, \boldsymbol{\theta}^n)}{\sigma^n} + \lambda_n \cdot h_n, \; 0 < \sigma^n \ll 1 . \tag{5}$$

The sign of $\delta_n := h_k^T \cdot s_n$ reveals if the adjusted Hessian approximation is negative definite. If $\delta_n \geq 0$ then the approximation is not negative definite, and λ_n is raised and s_n is estimated again. The new approximation s'_n can be derived from the old s_n via $s'_n = s_n + (\lambda_n - \lambda'_n) \cdot h_n$ where λ'_n is the raised λ_n. It is possible to determine in one step how much λ_n has to be risen to assure $\delta_n < 0$:

$$\delta_n < 0 \Leftrightarrow h_n^T \cdot s'_n < 0 \Leftrightarrow \delta_n + (\lambda_n - \lambda'_n) \cdot |h_n|^2 < 0 \Leftrightarrow \lambda'_n > \lambda_n + \delta_n / |h_n|^2 \ .$$

Following Møller, we chose $\lambda'_n = 2.0 \cdot \left(\lambda_n + \delta_n/|h_n|^2\right)$, i.e., $\delta'_n = -\delta_n - \lambda \cdot |h_n|^2$ as new estimate for δ_n.

This scaling mechanism combined with conjugate search directions as done in the CGEM (4) leads (in principle) to the SCGEM. For the sake of closeness to SCGs, we used in (5) the approximation of the second order information originally proposed by Møller. It requires one additional log-likelihood evaluation compared with the EM. Motivated by (1) and (3), we apply a different approximation in order to avoid the additional evaluation, namely $\nabla^2 LL(\mathbf{D}, \boldsymbol{\theta}^n) \approx E\left[\nabla^2 LL(\mathbf{Z} \mid \boldsymbol{\theta}) \mid \boldsymbol{\theta}^n, \mathbf{D}\right]$. It turns out that we approximate the Hessian of the log-likelihood $\nabla^2 LL(\mathbf{D}, \boldsymbol{\theta}^n)$ with the Hessian of the expected information score $\nabla^2 Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^n, \mathbf{D})$:

$$\begin{aligned}
& E\left[\frac{\partial^2 \ LL(\mathbf{Z} \mid \boldsymbol{\theta})}{\partial \beta_{i'j'k'} \partial \beta_{ijk}} \;\middle|\; \boldsymbol{\theta}^n, \mathbf{D}\right] \\
&= E\left[\frac{\partial}{\partial \beta_{i'j'k'}} \left(\mathrm{c}(\mathbf{fa}_i^{jk} \mid \mathbf{D}) - \theta_{ijk} \sum_l \mathrm{c}(\mathbf{fa}_i^{lk} \mid \mathbf{D})\right) \;\middle|\; \boldsymbol{\theta}^n, \mathbf{D}\right] \\
&= \left(-\sum_l \mathrm{ec}(\mathbf{fa}_i^{lk} \mid \mathbf{D})\right) \cdot \frac{\partial \theta_{ijk}}{\partial \beta_{i'j'k'}} \qquad (6) \\
&= \frac{\partial}{\partial \beta_{i'j'k'}} \left(\mathrm{ec}(\mathbf{fa}_i^{jk} \mid \mathbf{D}) - \theta_{ijk} \sum_l \mathrm{ec}(\mathbf{fa}_i^{lk} \mid \mathbf{D})\right) = \frac{\partial^2 Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^n, \mathbf{D})}{\partial \beta_{i'j'k'} \partial \beta_{ijk}} \ .
\end{aligned}$$

Now, recall that we are working in the reparameterized space, i.e., the β parameters are independent. The partial derivatives in (6) are zero when $i' \neq i$, $j' \neq j$, and $k' \neq k$. Thus, only the diagonal elements of $\nabla^2 Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^n, \mathbf{D})$ are non-zero. They are given by

$$\theta_{ijk} \cdot (\theta_{ijk} - 1) \cdot \sum_l \mathrm{ec}(\mathbf{fa}_i^{lk} \mid \mathbf{D}) \ .$$

Moreover, the computation of s_n becomes linear in the number of parameters, namely

$$s_n = \mathrm{diag}(\nabla^2 Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^n, \mathbf{D}))^T \cdot h_n + \lambda_n \cdot h_n \ . \qquad (7)$$

Because all involved quantities have been already computed for the gradient, the approximation does not cost any additional log-likelihood evaluations. Thus, SCGEM performs as many evaluations as the EM per iteration namely one.

6 Experiments

In the experiments described below, we implemented all three algorithms, i.e., EM, CGEM, and SCGEM using Netica API (`http://www.norsys.com`) for

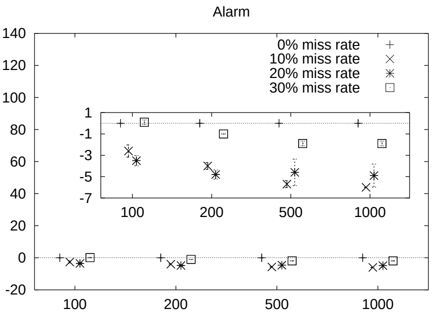
Table 1. Description of the networks used in the experiments

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 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 $3 \times 1 \times 3$ (resp. $5 \times 3 \times 5$) nodes. Nodes of the first and third layers have 3 possible states, nodes of the second 2. In total, there are 81 (resp. 568) parameters.

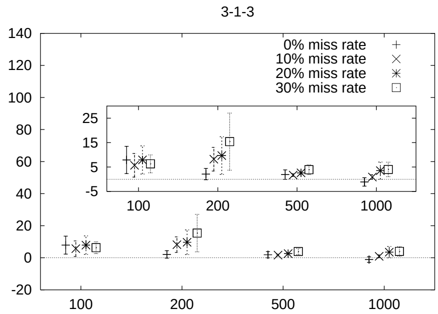
Bayesian network inference. We adapted the conjugate gradient described in [20] to fullfil the constraints described above and to become a CGEM. Based on this code, we adapted the scaled conjugate gradient as implemented in the Netlab library [16] to come up with the SCGEM. Two initial EM steps and a maximum of three consecutive scaling steps were set. To avoid zero entries in the associated conditional probability tables we used m-estimates ($m = 1$) [14].

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 (independently) 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 training sets were all subsets of the corresponding 1000 data sets. The values of latent nodes were never observed. For each training set, 10 different random initial sets of parameters were tried. We ran each algorithm on each data set starting from each of the initial sets of parameters. Each algorithm stopped when a limit of 200 iterations was exceeded or a change in log-likelihood at one iteration relative to the total change so far was smaller than $1 + 10^{-4}$. All learned models were evaluated on the test set using a standard measure, the *normalized-loss* $1/N \sum_{l=1}^{N} (\log P^*(\mathbf{d}_l) - \log P(\mathbf{d}_l))$, 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.

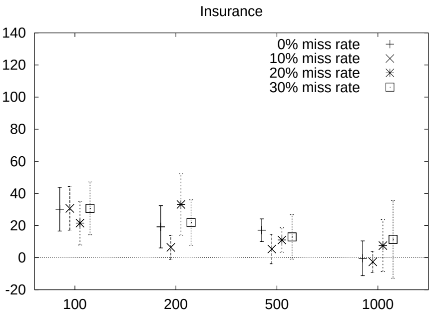
To measure the computational costs, we counted the number of log-likelihood evaluations. This way we do not need to compare CPU times, which depend on implementation details. More over, the evaluations domininate all other costs for sufficiently large number N of data cases (assuming unit costs of arithmetic operations). All three algorithms require the computation of the expected counts, i.e., a likelihood evaluation. Computing (4) takes $O(M)$ extra work, and computing (7) takes $O(M \cdot d)$ extra work where d is the maximal number of possible states of a random variable in a given Bayesian network of M parameters. Thus, the work needed to be done in each (line search) iteration is approximately the same. This complies with our experience gained in the experiments.



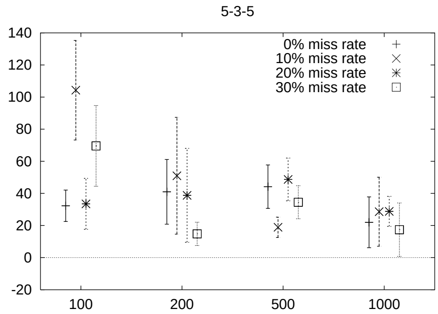
2	6	8	10	2	6	7	10	2	5	7	9	2	5	7	9
2	9	11	10	2	10	12	11	2	11	12	11	2	11	12	11



24	25	29	29	22	23	30	45	17	17	21	28	17	20	24	24
16	19	22	23	20	15	20	30	15	15	19	24	18	19	20	20



69	75	79	75	64	64	96	90	86	72	67	90	53	51	69	71
39	44	58	45	45	57	63	68	69	66	56	67	53	53	62	60



78	120	77	131	93	102	140	91	116	110	158	148	102	122	122	139
46	16	44	61	52	52	101	76	72	91	109	114	80	94	94	122

Fig. 1. The 95% confidence intervals for differences in number of iterations (EM−SCGEM). For each number of data cases (x axis) and each percentage of missing data (different line styles), the confidence intervals are shown. The figures within figures are close-ups of the corresponding areas. Each table below a figure shows the corresponding mean number of iterations for EM (upper row) and SCGEM (lower row) in order of appearance within the figure and rounded to the closest integer value.

The CGEM showed the expected behaviour. Considering only the normalized-loss, it reached similar normalized-losses than the EM. The 95% confidence (EM−CGEM) interval was $[-0.0175, -0.0099]$ favoring the EM. However, the CGEM had much higher computational costs than SCGEM. On average 19 line search iterations. To be fair, it has been argued that the CGEM does not need a very precise line search. For instance, Ortiz and Kaelbling [18] set a maximum of 10 line search iterations, and reported that this limit was often exceeded. To simulate this, we fixed the log-likelihoods for CGEM originally reached, but varied the averaged number of line search iterations. It turned out that SCGEM would run significantly faster than the CGEM already with a limit of 3 line search inferences (95% confidence interval CGEM−SCGEM for a limit of 3 would be $[22.42, 27.20]$, for a limit of 10 it would be $[115.98, 128.86]$) [3]. Therefore, we omit CGEM from further discussion.

[3] Note that this analysis actually favors the CGEM. It is unlikely that the CGEM would reach the original log-likelihoods within one line search.

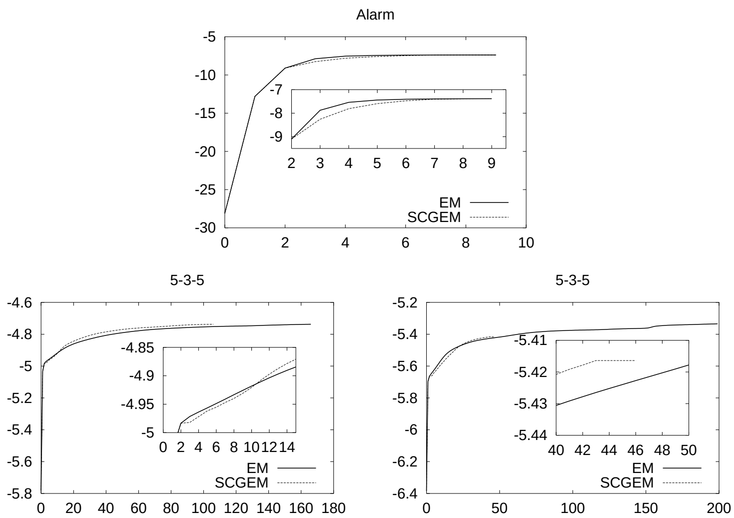


Fig. 2. Typical learning curves for EM and SCGEM on **Alarm** and **5-3-5**. The number of iterations, i.e., the number of log-likelihood evaluations is plotted against the log-likelihood achieved. The lower right plot shows the SCGEM getting trapped in scaling. The figures within figures are close-ups of the corresponding areas.

EM and SCGEM reached similar normalized-losses, i.e., were equal in quality. A *two-tailed, paired sampled t test* over all experiments showed that we cannot reject the null-hypothesis that EM and SCGEM differ on average performance ($p = 0.05$). There was a very small variation over the different networks. The confidence intervals (95%) for normalized-losses were: **Alarm** $[-0.00116, -0.00025]$, **Insurance** $[-0.00848, +0.00120]$, **3-1-3** $[-0.00559, +0.00014]$ and **5-3-5** $[+0.00591, +0.1620]$.

In total, SCGEM was faster then EM. Figure 1 summarizes the difference (EM$-$SCG) in number of iterations, i.e., number of likelihood iterations. The 95% confidence intervals are plotted. Readably, EM was faster than SCGEM on **Alarm**. However, there was only a difference of around 6 iterations. Moreover, this did not carry over to the other domains. SCGEM tended to be slightly faster on **3-1-3** (≈ 5 iterations speed-up on average). On **Insurance** and **5-3-5** , the difference in average iterations increased to a ≈ 16 speed-up for **Insurance** and a ≈ 39 speed-up for **5-3-5** . However, this is averaged over different percentages of missing data. As Figure 1 shows, there can be speed-ups of more than 80 iterations depending on the percentage of missing informations. These results validate the theory as explained in Section 4: the more missing information, the higher the speed-up.

Compared with SCG, Kersting and Landwehr [9] reported that the EM had a faster initial progress. Compared to SCGEM, this does not hold any longer. Figure 2 shows some typical learning curves. For **Alarm**, the EM typically possessed a faster initial progress, but e.g. for **5-3-5**, the SCGEM typically overtook the

EM after few iterations. However in contrast to the EM, the SCGEM got sometimes trapped in scaling. Surprisingly, this happened most of time for **Alarm** (54% of 136 cases) where SCGEM got no speed-up. The remaining trapped cases were distributed as follows: 6% on **3-1-3**, 16% on **Insurance**, and 24% on **5-3-5**. Interestingly, SCGEM reached better normalized-losses on **5-3-5** (taking the 'scaling traps' into account).

7 Related Work and Conclusions

Both, EM and gradients are well-known parameter estimation techniques. For a general introduction see [13, 12]. Møller [15] introduced SCG to estimate the parameters of neural networks. Lauritzen [11] introduced EM for Bayesian networks (see also Heckerman [6] for a nice tutorial). The original work on gradient-based approaches in the context of Bayesian networks was done by Binder *et al.* [3]. However, we are not aware of any application of SCG in order to accelerate the EM. Bauer *et al.* [1] reported on experiments with PEM for learning Bayesian networks. They did not report on results of CGEM. Thiesson [21] discussed conventional conjugate gradient accelerations of the EM, but did not report on experiments. Ortiz and Kaelbling [17] conducted experiments with PEM and CGEM for continuous models, namely density estimation with a mixture of Gaussian. Their results generally favor CGEM over PEM, although PEM can be superior for some domains. Finally, there are several acceleration techniques of the EM which do not apply conjugate gradients and have not reached much attention within Bayesian network learning. We refer to [8, 13] for a full account of them. Among this work, Lange [10] is closest to SCGEM. He proposed the expected information matrix as approximation of the Hessian together with a symmetric, rank-one update of the *complete* approximation matrix within a quasi-Newton technique. An interesting but orthogonal research discussed in [13] investigates criterions when to switch to CGEM. This is a promising research direction for SCGEM.

To conclude, we introduced *scaled conjugate gradient* EM for maximum likelihood parameter estimation of Bayesian networks. They overcome the expensive line search of traditional conjugate gradient EM. To the best of our knowledge, it is the first time that one reports on an accelerated EM for Bayesian networks which exhibits the same number of likelihood estimations per iteration then the EM. The experiments show that SCGEM is equal with EM and CGEM in quality but can significantly accelerate both. As predicted by theory, SCGEM seems especially well suited in the presence of many latent parameters, i.e., 'EM-hard' instances. The main future question seems to be the disarming of the scaling traps.

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 *LNMI*, 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. G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.
5. 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.
6. D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, 1995.
7. M. Jamshidian and R. I. Jennrich. Conjugate Gradient Accleration of the EM Algorithm. *Journal of the American Statistical Association*, 88(412):221–228, 1993.
8. M. Jamshidian and R. I. Jennrich. Accleration of the EM Algorithm by using Quasi-Newton Methods. *Jour. of the Royal Stat. Society B*, 59(3):569–587, 1997.
9. K. Kersting and N. Landwehr. Scaled Conjugate Gradients for Maximum likelihood: An Empirical Comparison with the EM Algorithm. In J. A. Gámez and A. Salmerón, editors, *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM-02)*, pages 89–98, Cuenca, Spain, 2002.
10. K. Lange. A quasi-Newton acceleration of the EM algorithm. *Statistica Sinica*, 5:1–18, 1995.
11. S. L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.
12. D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 1984.
13. G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, 1997.
14. T. M. Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc., 1997.
15. M. Møller. A Scaled Conjugate Gradient Algoritm for Fast Supervised Learning. *Neural Networks*, 6:525–533, 1993.
16. I. Nabney. *NETLAB: Algorithms for Pattern Recognition*. Advances in Pattern Recognition. Springer-Verlag, 2001. `http://www.ncrg.aston.ac.uk/netlab/`.
17. 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.
18. L. E. Ortiz and L. P. Kaelbling. Notes on methods based on maximum-likelihodd estimation for learning the parameters of the mixture-of-Gaussians model. Technical Report CS-99-03, Department of Computer Science, Brown University, 1999.
19. J. Pearl. *Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2. edition, 1991.
20. 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`.
21. 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.