A Gradient-Based Algorithm Competitive with Variational Bayesian EM for Mixture of Gaussians

Mikael Kuusela, Tapani Raiko, Antti Honkela, and Juha Karhunen

***Abstract*— While variational Bayesian (VB) inference is typically done with the so called VB EM algorithm, there are models where it cannot be applied because either the E-step or the M-step cannot be solved analytically. In 2007, Honkela et al. introduced a recipe for a gradient-based algorithm for VB inference that does not have such a restriction. In this paper, we derive the algorithm in the case of the mixture of Gaussians model. For the first time, the algorithm is experimentally compared to VB EM and its variant with both artificial and real data. We conclude that the algorithms are approximately as fast depending on the problem.**

I. Introduction

Variational Bayesian (VB) inference (see e.g. [3]) is a tool for machine learning of probabilistic models that is more accurate than traditional point estimates (least squares, maximum likelihood, maximum a posteriori) but still very fast compared to sampling (MCMC) methods. VB is especially useful with latent variable models where the number of unknown variables is often very large which makes point estimates overfit on one hand, and sampling methods very slow on the other. VB takes into account the uncertainty of the unknown variables by estimating a probability distribution q for them. In this paper, we use a parameterized distribution q.

The standard way to do VB inference is the VB expectation-maximization (EM) algorithm. While the VB EM algorithm provides a straightforward way of learning models in the conjugate-exponential family, there are more complicated models for which the VB EM algorithm is not available such as the nonlinear state-space model (NSSM) of [14]. In such cases, it is still possible to compute (or at least approximate) the VB cost as a function of the variational parameters $\boldsymbol{\xi}$ and minimize using a suitable optimization method. Recently in [6], such a method was tailored for VB, but it has not been compared to the VB EM algorithm. This paper does the comparison using the mixture of Gaussians (MoG) problem which is known to be well-fitted for the VB EM algorithm.

We show in this paper that a gradient-based algorithm can be made competitive with the VB EM algorithm in computation time. Each update is more costly, but the number of required iterations before convergence is lower. Having more than one algorithm for the same problem can be useful in many ways. Different algorithms end up in different locally optimal solutions. Some algorithms are easier to parallelize and become more efficient with massively parallel hardware. Also, some algorithms are easier to be implemented locally, that is, they could be implemented as neural systems. Also, gradient-based algorithms can trivially be used for online learning.

The rest of the paper is as follows. Section II gives an introduction to VB and the VB EM algorithm. Section III describes gradient-based algorithms for VB. Section IV describes the variational mixture of Gaussians model and how the different algorithms are implemented for it. Section V gives experimental results with both artificial data and natural images, after which we conclude.

II. Variational Bayesian Inference

Many machine learning problems can be seen as inferring the parameters $\boldsymbol{\theta}$ and the latent variables $\mathbf{Z}$ of some model given the observed data $\mathbf{X}$, where the latent variables are quantities related to each data sample that cannot be observed directly. Thus, the number of latent variables is proportional to the number of observations, while the parameters $\boldsymbol{\theta}$ are shared between all data samples. The Bayes' rule can be used for this inference task in the form

$$p(\boldsymbol{\theta}, \mathbf{Z}|\mathbf{X}) = \frac{p(\mathbf{X}|\boldsymbol{\theta}, \mathbf{Z})p(\boldsymbol{\theta}, \mathbf{Z})}{p(\mathbf{X})} \tag{1}$$

which gives us the posterior probability distribution of unknowns $\boldsymbol{\theta}$ and $\mathbf{Z}$.

The prior $p(\boldsymbol{\theta}, \mathbf{Z})$ in Equation (1) can be interpreted as our knowledge of the model parameters *before* the data $\mathbf{X}$ is observed while the posterior $p(\boldsymbol{\theta}, \mathbf{Z}|\mathbf{X})$ gives us the parameter distribution *after* the data is observed. Thus, the observation of the data can be seen as changing our prior beliefs about the parameters.

The evidence $p(\mathbf{X})$ in (1) can be evaluated to be

$$p(\mathbf{X}) = \int_{\boldsymbol{\theta}, \mathbf{Z}} p(\mathbf{X}, \boldsymbol{\theta}, \mathbf{Z}) d\boldsymbol{\theta} d\mathbf{Z}. \tag{2}$$

The central issue in Bayesian inference is that, apart from the simplest models, this integral is intractable. Also, for example the evaluation of the predictive distribution $p(\mathbf{y}|\mathbf{X})$, that is the distribution of a new observation $\mathbf{y}$ given the observed data $\mathbf{X}$, becomes intractable as it requires integration over the posterior distribution.

Variational methods attempt to overcome intractable integrals such as (2) by approximating the true posterior distribution $p(\boldsymbol{\theta}, \mathbf{Z}|\mathbf{X})$ by another distribution $q(\boldsymbol{\theta}, \mathbf{Z})$. We use the Kullback–Leibler divergence to measure the misfit

The authors are with the Adaptive Informatics Research Center, Helsinki University of Technology (TKK), Helsinki, Finland (email: firstname.lastname@tkk.fi)

between these distributions, defined as

$$
\begin{aligned}
D_{KL}(q||p) &= \int_{\boldsymbol{\theta},\mathbf{Z}} q(\boldsymbol{\theta},\mathbf{Z}) \ln \frac{q(\boldsymbol{\theta},\mathbf{Z})}{p(\boldsymbol{\theta},\mathbf{Z}|\mathbf{X})} d\boldsymbol{\theta} d\mathbf{Z} \\
&= E_q \left\{ \ln \frac{q(\boldsymbol{\theta},\mathbf{Z})}{p(\boldsymbol{\theta},\mathbf{Z}|\mathbf{X})} \right\}
\end{aligned} \tag{3}
$$

where $E_q\{\cdot\}$ denotes the expectation over distribution q.

As the true posterior $p(\boldsymbol{\theta},\mathbf{Z}|\mathbf{X})$ in (3) is unknown, we will subtract the log-evidence, which is a constant, from this to obtain the true cost function $\mathcal{C}$ used in the learning process:

$$
\mathcal{C} = D_{KL}(q||p) - \ln p(\mathbf{X}) = E_q \left\{ \ln \frac{q(\boldsymbol{\theta},\mathbf{Z})}{p(\boldsymbol{\theta},\mathbf{Z},\mathbf{X})} \right\}. \tag{4}
$$

Because the Kullback–Leibler divergence is non-negative, the negative cost function $-\mathcal{C}$ is a lower bound on the log model evidence $\ln p(\mathbf{X})$.

To make the integral in (4) tractable one has to somehow restrict the form of the distribution $q(\boldsymbol{\theta},\mathbf{Z})$. One way to accomplish this is to select the functional form of the distribution $q(\boldsymbol{\theta},\mathbf{Z})$ governed by some parameters $\boldsymbol{\xi}$ which we will denote by $q(\boldsymbol{\theta},\mathbf{Z}|\boldsymbol{\xi})$. A popular choice for this fixed form solution is a factorized Gaussian distribution.

VB learning has recently become popular in inference tasks due to its capability to automatically select the complexity of the model and mostly avoid overfitting the data while still being computationally efficient enough to be able to solve real world problems.

A. The VB EM Algorithm

Let us assume that the approximate posterior $q(\boldsymbol{\theta},\mathbf{Z})$ will factorize between the parameters $\boldsymbol{\theta}$ and the latent variables $\mathbf{Z}$, that is

$$
q(\boldsymbol{\theta},\mathbf{Z}) = q(\boldsymbol{\theta})q(\mathbf{Z}). \tag{5}
$$

The VB EM algorithm alternates between two update steps, the E-step finds the optimal $q(\mathbf{Z})$ assuming that $q(\boldsymbol{\theta})$ is fixed, and the M-step finds the optimal $q(\boldsymbol{\theta})$ assuming that $q(\mathbf{Z})$ is fixed. It is shown for example in [3] that the solutions are

$$
q(\mathbf{Z}) \propto \exp(E_{q(\boldsymbol{\theta})}\{\ln p(\boldsymbol{\theta},\mathbf{Z},\mathbf{X})\}) \tag{6}
$$

$$
q(\boldsymbol{\theta}) \propto \exp(E_{q(\mathbf{Z})}\{\ln p(\boldsymbol{\theta},\mathbf{Z},\mathbf{X})\}). \tag{7}
$$

These steps are repeated until convergence is achieved, which is determined by evaluating the cost function (4) on each iteration.

It was proposed in [7] that cyclic parameter update algorithms such as VB EM could be accelerated using a technique called pattern searches. The idea is that after every few cyclic updates, a line search is performed in the direction defined by the difference of the current estimate of the unknowns and the old estimate one update cycle before that.

III. Gradient-Based Learning Algorithms

The VB cost function $\mathcal{C} = \mathcal{C}(\boldsymbol{\xi})$ in (4) is a function of the variational parameters $\boldsymbol{\xi}$ that define the distribution q once the functional form of q is fixed. As a results, VB inference can also be done using standard nonlinear optimization techniques to find the minimum of $\mathcal{C}$.

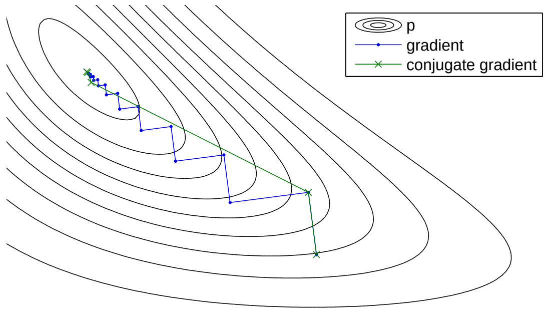


Fig. 1. Gradient and conjugate gradient updates are applied to finding the maximum of the posterior $p(x,y) \propto \exp[-9(xy-1)^2 - x^2 - y^2]$. The step sizes that maximize p are used. Note that the first steps are the same, but the following gradient updates are orthogonal whereas conjugate gradient finds a much better direction.

A. Gradient Descent

The simplest nonlinear optimization technique is the gradient descent. In that method, one first evaluates the negative gradient of the cost function $\mathbf{p}_k = -\nabla\mathcal{C}(\boldsymbol{\xi}_k)$, then performs a line search in the direction of $\mathbf{p}_k$ to obtain a step size λ and finally updates the parameters using this step size $\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + \lambda\mathbf{p}_k$. Instead of line search, the step size λ can also be set to some sufficiently small constant or adjusted adaptively.

B. Conjugate Gradient Descent

There are many speed-ups to the basic gradient descent algorithm one of which is the conjugate gradient (CG) method. Figure 1 shows a comparison of gradient and conjugate gradient updates in a simple problem.

In the CG method, the search direction is set to the negative of the gradient on the first iteration just like in gradient descent but on subsequent iterations, the search direction is

$$
\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1} \tag{8}
$$

where $\mathbf{g}_k = \nabla\mathcal{C}(\boldsymbol{\xi}_k)$, $\mathbf{p}_{k-1}$ is the previous search direction and β_k can be calculated using the Polak-Ribiére formula [10]

$$
\beta_k = \frac{\langle(\mathbf{g}_k - \mathbf{g}_{k-1}), \mathbf{g}_k\rangle}{\|\mathbf{g}_{k-1}\|^2} = \frac{(\mathbf{g}_k - \mathbf{g}_{k-1})^T\mathbf{g}_k}{\mathbf{g}_{k-1}^T\mathbf{g}_{k-1}}. \tag{9}
$$

The convergence of the algorithm can only be guaranteed if β_k is restricted to be non-negative [13]. In practice, we reset the search direction $\mathbf{p}_k$ to the negative gradient each time the Polak-Ribiére formula returns a negative β_k.

C. Natural Gradient Descent

Natural gradient descent [2] is based on differential geometry where the geometry of the parameter space is not Euclidean. Changing the parameters by a certain amount might have a relatively small or large effect on the distribution that the parameters define, and the natural gradient can take this difference into account. Figure 2 shows a comparison of gradient and natural gradient directions in a simple problem.

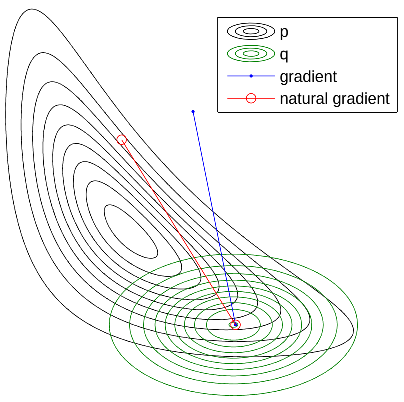


Fig. 2. Gradient and natural gradient directions are shown for the mean of distribution q. VB learning with a diagonal covariance is applied to the posterior $p(x,y) \propto \exp[-9(xy-1)^2 - x^2 - y^2]$. The natural gradient strengthens the updates in the directions where the uncertainty is large.

In a Riemannian manifold S, the inner product is given by

$$\langle \mathbf{v}, \mathbf{u} \rangle_p = \mathbf{v}^T \mathbf{G} \mathbf{u} \tag{10}$$

where $\mathbf{G} = (g_{ij})$ is called the Riemannian metric tensor of the manifold S at point p [9]. As a consequence of (10), the squared norm of vector $\mathbf{v}$ in a Riemannian manifold is

$$\|\mathbf{v}\|^2 = \langle \mathbf{v}, \mathbf{v} \rangle_p = \mathbf{v}^T \mathbf{G} \mathbf{v} \tag{11}$$

which is analogous with the squared norm in Euclidean space

$$\|\mathbf{v}\|^2 = \langle \mathbf{v}, \mathbf{v} \rangle = \mathbf{v}^T \mathbf{v}. \tag{12}$$

Using information geometry, the parameter space $\boldsymbol{\xi}$ of probability distributions $q(\boldsymbol{\theta}, \mathbf{Z}|\boldsymbol{\xi})$ can be regarded as a Riemannian manifold whose Riemannian metric tensor $\mathbf{G}$ is given by the Fisher information matrix [1], [9]

$$g_{ij}(\boldsymbol{\xi}) = E_q \left\{ \frac{\partial \ln q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_i} \frac{\partial \ln q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_j} \right\}. \tag{13}$$

If the geometry of the parameter space of $\mathcal{C}(\boldsymbol{\xi})$ is considered Riemannian, the direction of the steepest ascent is given, instead of the gradient, by the natural gradient [2]

$$\tilde{\nabla} \mathcal{C}(\boldsymbol{\xi}) = \mathbf{G}^{-1}(\boldsymbol{\xi}) \nabla \mathcal{C}(\boldsymbol{\xi}). \tag{14}$$

Thus, the gradient descent method of section III-A becomes

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k - \lambda \tilde{\nabla} \mathcal{C}(\boldsymbol{\xi}_k). \tag{15}$$

This is called the natural gradient descent algorithm.

Original applications [2] of the natural gradient were based on the geometry of $p(\mathbf{X}|\boldsymbol{\theta})$ to update the point estimates of the model parameters $\boldsymbol{\theta}$. For applications to learning MLP networks, it was proposed [5] that the dependencies between weights in different layers would be ignored, making the approximate matrix $\mathbf{G}$ block diagonal and hence easier to invert, since only the separate blocks need to be inverted.

Our approach, proposed in [6], is that natural gradient could be applied to update both the model parameters and the latent variables by using the geometry of the variational Bayesian approximation $q(\boldsymbol{\theta}, \mathbf{Z}|\boldsymbol{\xi})$. The matrix inversion required for the evaluation of the natural gradient in (14) would be prohibitively expensive if the full matrix had to be inverted. Luckily, because of the typical factorizing approximation of q, the matrix $\mathbf{G}$ is block diagonal [6] without further approximations.

D. Natural Conjugate Gradient Descent

The natural and conjugate gradient methods can be combined by using conjugate directions as described in Section III-B by replacing the gradient in (8) with the natural gradient

$$\mathbf{g}_k \hookrightarrow \tilde{\mathbf{g}}_k = \tilde{\nabla} \mathcal{C}(\boldsymbol{\xi}). \tag{16}$$

It was shown to lead to faster convergence and better minima than the standard CG descent in MLP training [4], and to significantly improve the performance in learning a nonlinear state-space model [6].

Note that the vector operations in (9) are performed in the Riemannian sense, using Equations (10) and (11). The costly vector-matrix multiplications can be avoided by noting that

$$\|\tilde{\mathbf{g}}_k\|^2 = \tilde{\mathbf{g}}_k^T \mathbf{G}_k \tilde{\mathbf{g}}_k = \tilde{\mathbf{g}}_k^T \mathbf{G}_k \mathbf{G}_k^{-1} \mathbf{g}_k = \tilde{\mathbf{g}}_k^T \mathbf{g}_k, \tag{17}$$

resulting in

$$\beta_k = \frac{\langle (\tilde{\mathbf{g}}_k - \tilde{\mathbf{g}}_{k-1}), \tilde{\mathbf{g}}_k \rangle}{\|\tilde{\mathbf{g}}_{k-1}\|^2} = \frac{(\tilde{\mathbf{g}}_k - \tilde{\mathbf{g}}_{k-1})^T \mathbf{g}_k}{\tilde{\mathbf{g}}_{k-1}^T \mathbf{g}_{k-1}}. \tag{18}$$

We call this combination the natural conjugate gradient (NCG) descent.

IV. Variational Mixture of Gaussians

The mixture of Gaussians is a probability distribution which is a linear combination of K Gaussian distributions [3]

$$p(\mathbf{x}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{19}$$

where $\mathbf{x}$ is a D-dimensional random variable and $\boldsymbol{\pi} = [\pi_1 \cdots \pi_K]^T$ are called the mixing coefficients while $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are the mean and the covariance matrix of the kth Gaussian component. The inverse of the covariance matrix $\boldsymbol{\Lambda}_k = \boldsymbol{\Sigma}_k^{-1}$ is called the precision matrix.

In the case of the MoG model, the latent variables $\mathbf{Z}$ discussed in Section II-A are the information on which one of the K Gaussian components has generated a particular observation $\mathbf{x}_n$. This information will be represented with a K-dimensional binary vector $\mathbf{z}_n$ whose elements z_{nk} are either 0 or 1 where 1 denotes the component responsible for generating the observed data point $\mathbf{x}_n$ in question. It should be noted that only one component can be responsible for generating a single observation and thus the elements of the vector $\mathbf{z}_n$ sum to unity. Let N denote the total number of observed data points. Now, all the N latent variables of the model can be regarded as forming a latent variable matrix $\mathbf{Z} = (z_{nk})$ of the order $N \times K$.

Given the mixing coefficients $\boldsymbol{\pi}$, the probability distribution over the latent variables is given by

$$p(\mathbf{Z}|\boldsymbol{\pi}) = \prod_{n=1}^{N}\prod_{k=1}^{K} \pi_k^{z_{nk}}. \tag{20}$$

As we want to use conjugate priors in our treatment, we next introduce a Dirichlet prior for the mixing coefficients

$$p(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}_0) \tag{21}$$

where $\boldsymbol{\alpha}_0 = [\alpha_0 \cdots \alpha_0]^T$ is a K-dimensional hyperparameter vector whose elements are all given by α_0 due to symmetry.

Similarly, the distribution over the data $\mathbf{X}$ given the latent variables $\mathbf{Z}$, the means $\boldsymbol{\mu}$ and the precision matrices $\boldsymbol{\Lambda}$ can be written as

$$p(\mathbf{X}|\mathbf{Z},\boldsymbol{\mu},\boldsymbol{\Lambda}) = \prod_{n=1}^{N}\prod_{k=1}^{K} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k^{-1})^{z_{nk}}. \tag{22}$$

Note that we are assuming here that the data vectors $\mathbf{x}_n$ are independent and identically distributed. In this case, the conjugate prior for the component parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}$ is given by the Gaussian-Wishart distribution

$$\begin{aligned} p(\boldsymbol{\mu},\boldsymbol{\Lambda}) &= p(\boldsymbol{\mu}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda}) \\ &= \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_0,(\beta_0\boldsymbol{\Lambda}_k)^{-1})\mathcal{W}(\boldsymbol{\Lambda}_k|\mathbf{W}_0,\nu_0). \end{aligned} \tag{23}$$

The joint distribution over all the random variables of the model is then given by

$$p(\mathbf{X},\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}) = p(\mathbf{X}|\mathbf{Z},\boldsymbol{\mu},\boldsymbol{\Lambda})p(\mathbf{Z}|\boldsymbol{\pi})p(\boldsymbol{\pi})p(\boldsymbol{\mu}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda}). \tag{24}$$

A. VB EM for the Mixture of Gaussians Model

The VB EM treatment of the mixture of Gaussians model described here is completely based on [3]. Because of this, some details of the derivation will be omitted here and we will concentrate only on the most important results.

We now make the factorizing approximation described by Equation (5)

$$q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}) = q(\mathbf{Z})q(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}) \tag{25}$$

and use Equations (6) and (7) along with Equation (24) to first update $q(\mathbf{Z})$ (E-step) and subsequently update $q(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})$ (M-step). The resulting approximate posterior distributions are

$$q(\mathbf{Z}) = \prod_{n=1}^{N}\prod_{k=1}^{K} r_{nk}^{z_{nk}} \tag{26}$$

and

$$q(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}) = q(\boldsymbol{\pi})q(\boldsymbol{\mu},\boldsymbol{\Lambda}) = q(\boldsymbol{\pi})\prod_{k=1}^{K} q(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k) \tag{27}$$

where

$$q(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \tag{28}$$

$$q(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k) = \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_k,(\beta_k\boldsymbol{\Lambda}_k)^{-1})\mathcal{W}(\boldsymbol{\Lambda}_k|\mathbf{W}_k,\nu_k). \tag{29}$$

In expressing the update rules for the distribution parameters in Equations (26), (28) and (29), we will find the following definitions useful:

$$N_k = \sum_{n=1}^{N} r_{nk} \tag{30}$$

$$\overline{\mathbf{x}}_k = \frac{1}{N_k}\sum_{n=1}^{N} r_{nk}\mathbf{x}_n \tag{31}$$

$$\mathbf{S}_k = \frac{1}{N_k}\sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \overline{\mathbf{x}}_k)(\mathbf{x}_n - \overline{\mathbf{x}}_k)^T \tag{32}$$

$$\ln\tilde{\Lambda}_k = \sum_{i=1}^{D}\psi\left(\frac{\nu_k+1-i}{2}\right) + D\ln 2 + \ln|\mathbf{W}_k| \tag{33}$$

$$\ln\tilde{\pi}_k = \psi(\alpha_k) - \psi(\sum_{k'=1}^{K}\alpha_{k'}) \tag{34}$$

where D is the dimensionality of the data and $\psi(\cdot)$ is the digamma function which is defined as the derivative of the log of the gamma function.

Using these definitions, the parameters r_{nk} of the approximate posterior over latent variables $q(\mathbf{Z})$ which are updated in the E-step are given by

$$r_{nk} = \frac{\rho_{nk}}{\sum_{l=1}^{K}\rho_{nl}} \tag{35}$$

where

$$\rho_{nk} = \tilde{\pi}_k\tilde{\Lambda}_k^{1/2}\exp\left(-\frac{D}{2\beta_k} - \frac{\nu_k}{2}(\mathbf{x}_n-\mathbf{m}_k)^T\mathbf{W}_k(\mathbf{x}_n-\mathbf{m}_k)\right). \tag{36}$$

The parameters r_{nk} are called *responsibilities* because they represent the responsibility the kth component takes in explaining the nth observation. The responsibilities are non-negative and their sum over k is one. They can be arranged into a matrix $\mathbf{R} = (r_{nk})$.

The parameter update equations for the M-step are then given by

$$\alpha_k = \alpha_0 + N_k \tag{37}$$

$$\beta_k = \beta_0 + N_k \tag{38}$$

$$\nu_k = \nu_0 + N_k \tag{39}$$

$$\mathbf{m}_k = \frac{1}{\beta_0 + N_k}(\beta_0\mathbf{m}_0 + N_k\overline{\mathbf{x}}_k) \tag{40}$$

$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + N_k\mathbf{S}_k + \frac{\beta_0 N_k}{\beta_0+N_k}(\overline{\mathbf{x}}_k-\mathbf{m}_0)(\overline{\mathbf{x}}_k-\mathbf{m}_0)^T. \tag{41}$$

B. The Cost Function

We can use Equation (4) along with Equations (20)-(29) to evaluate the cost function for the learning process

$$\begin{aligned} \mathcal{C} =& \sum_{\mathbf{Z}}\int_{\boldsymbol{\pi}}\int_{\boldsymbol{\mu}}\int_{\boldsymbol{\Lambda}} q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})\ln\frac{q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})}{p(\mathbf{X},\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})}d\boldsymbol{\pi}d\boldsymbol{\mu}d\boldsymbol{\Lambda} \\ =& E_q\left\{\ln q(\mathbf{Z}) - \ln p(\mathbf{Z}|\boldsymbol{\pi})\right\} + E_q\left\{\ln q(\boldsymbol{\pi}) - \ln p(\boldsymbol{\pi})\right\} \\ &+ E_q\left\{\ln q(\boldsymbol{\mu},\boldsymbol{\Lambda}) - \ln p(\boldsymbol{\mu},\boldsymbol{\Lambda})\right\} - E_q\left\{\ln p(\mathbf{X}|\mathbf{Z},\boldsymbol{\mu},\boldsymbol{\Lambda})\right\} \end{aligned} \tag{42}$$

These expectations can be evaluated to give [3]

$$E_q\left\{\ln q(\mathbf{Z}) - \ln p(\mathbf{Z}|\boldsymbol{\pi})\right\} = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk} \ln \frac{r_{nk}}{\tilde{\pi}_k} \tag{43}$$

$$E_q\left\{\ln q(\boldsymbol{\pi}) - \ln p(\boldsymbol{\pi})\right\} = \sum_{k=1}^{K} (\alpha_k - \alpha_0) \ln \tilde{\pi}_k + \ln \frac{C(\boldsymbol{\alpha})}{C(\boldsymbol{\alpha}_0)} \tag{44}$$

$$\begin{aligned} &E_q\left\{\ln q(\boldsymbol{\mu}, \boldsymbol{\Lambda}) - \ln p(\boldsymbol{\mu}, \boldsymbol{\Lambda})\right\} = \\ &\frac{1}{2}\sum_{k=1}^{K}\Big\{ D\Big(\frac{\beta_0}{\beta_k} - 1 + \ln \frac{\beta_k}{\beta_0}\Big) - H_q\{\boldsymbol{\Lambda}_k\} \\ &- \beta_0 \nu_k (\mathbf{m}_k - \mathbf{m}_0)^T \mathbf{W}_k (\mathbf{m}_k - \mathbf{m}_0) + \nu_k \operatorname{Tr}(\mathbf{W}_0^{-1}\mathbf{W}_k)\Big\} \\ &- K \ln B(\mathbf{W}_0, \nu_0) - \frac{\nu_0 - D - 1}{2} \sum_{k=1}^{K} \ln \tilde{\Lambda}_k \end{aligned} \tag{45}$$

$$\begin{aligned} &E_q\left\{\ln p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})\right\} = \\ &\frac{1}{2}\sum_{k=1}^{K} N_k \Big\{ \ln \tilde{\Lambda}_k - \frac{D}{\beta_k} - \nu_k \operatorname{Tr}(\mathbf{S}_k \mathbf{W}_k) \\ &- \nu_k (\overline{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{m}_k) - D \ln 2\pi \Big\} \end{aligned} \tag{46}$$

where $\operatorname{Tr}(\mathbf{A})$ denotes the trace of matrix $\mathbf{A}$ and $H_q\{\boldsymbol{\Lambda}_k\}$ is the entropy of the distribution $q(\boldsymbol{\Lambda}_k)$.

The cost function $\mathcal{C}$ given by Equation (42) can be used to determine when the VB EM algorithm has converged. The cost function will decrease during each iteration and when the difference between the previous cost function value $\mathcal{C}_{k-1}$ and the current value $\mathcal{C}_k$ becomes sufficiently small we can assume that the learning process has converged.

C. Natural Conjugate Gradient for Mixture of Gaussians

To be able to compare the VB EM and NCG algorithms, we assume that the approximate posterior distribution $q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})$ takes the same functional form as in the case of the VB EM algorithm. Thus, the fixed form posterior distributions are given by Equations (26), (28) and (29) and the cost function which is to be minimized by the NCG algorithm is given Equation (42). In this work, we optimize only the responsibilities r_{nk} and the means $\mathbf{m}_k$ using gradient-based methods. All the other model parameters, namely the parameters α_k of the Dirichlet distribution, the parameters β_k controlling the covariance of the component means as well as the parameters $\mathbf{W}_k$ and ν_k of the Wishart distribution, are updated using the VB EM update Equations (37), (38), (39) and (41).

There are a few things that have to be taken into account when deriving gradient-based algorithms for the mixture of Gaussians model. Firstly, using the *softmax* parametrization

$$r_{nk} = \frac{e^{\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl}}}, \tag{47}$$

it can be easily seen that the responsibilities are always positive and $\sum_{k=1}^{K} r_{nk} = 1$. Secondly, if we set the responsibilities $r_{nk}, n = 1 \ldots N, k = 1 \ldots K-1$ to some values, the values of $r_{nK}, n = 1 \ldots N$ must be $r_{nK} = 1 - \sum_{k=1}^{K-1} r_{nk}$. As a result, the number of degrees of freedom in the responsibilities of the model is not the number of responsibilities NK but instead $N(K-1)$. When we are using the parametrization (47), this means that we can regard the parameters γ_{nK} as constants and only optimize the cost function with respect to parameters $\gamma_{nk}, n = 1 \ldots N, k = 1 \ldots K-1$. This is especially important when using the natural gradient.

The gradient of the cost function (42) with respect to $\mathbf{m}_k$ is given by

$$\nabla_{\mathbf{m}_k} \mathcal{C} = \nu_k \mathbf{W}_k (N_k(\mathbf{m}_k - \overline{\mathbf{x}}_k) + \beta_0(\mathbf{m}_k - \mathbf{m}_0)), k = 1 \ldots K \tag{48}$$

and the derivative with respect to γ_{nk} is given by

$$\frac{\partial \mathcal{C}}{\partial \gamma_{nk}} = E_{nk} - r_{nk} F_n, n = 1 \ldots N, k = 1 \ldots K-1 \tag{49}$$

where

$$\begin{aligned} E_{nk} = r_{nk}\Big(\ln r_{nk} - \ln \tilde{\pi}_k - \frac{1}{2}\Big(\ln \tilde{\Lambda}_k \\ - \frac{D}{\beta_k} - D \ln 2\pi - \nu_k (\mathbf{x}_n - \mathbf{m}_k)^T \mathbf{W}_k (\mathbf{x}_n - \mathbf{m}_k)\Big)\Big) \end{aligned} \tag{50}$$

and

$$F_n = \sum_{k=1}^{K} E_{nk}. \tag{51}$$

We can update the responsibilities r_{nk} without having to evaluate and store the parameters γ_{nk} by noting that

$$\begin{aligned} r'_{nk} &= \frac{e^{\gamma_{nk} + \Delta\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl} + \Delta\gamma_{nl}}} \\ &= \frac{\sum_{l=1}^{K} e^{\gamma_{nl}}}{\sum_{l=1}^{K} e^{\gamma_{nl} + \Delta\gamma_{nl}}} \frac{e^{\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl}}} e^{\Delta\gamma_{nk}} = c_n r_{nk} e^{\Delta\gamma_{nk}} \end{aligned} \tag{52}$$

where r'_{nk} is the new responsibility, $\Delta\gamma_{nk}$ is the change in parameter γ_{nk} determined by line search in the direction of the negative gradient and c_n is a normalizing constant which makes sure that $\sum_{k=1}^{K} r'_{nk} = 1$. Thus c_n can also be expressed in the form $c_n = (\sum_{k=1}^{K} r_{nk} e^{\Delta\gamma_{nk}})^{-1}$ and we can update the responsibilities using the formula

$$r'_{nk} = \frac{r_{nk} e^{\Delta\gamma_{nk}}}{\sum_{l=1}^{K} r_{nl} e^{\Delta\gamma_{nl}}}. \tag{53}$$

In order to use the natural gradient, we need to know the Riemannian metric tensor $\mathbf{G}$ of the parameter space $(\mathbf{m}, \boldsymbol{\gamma})$. The matrix $\mathbf{G}$ is given by Equation (13). The resulting matrix is a block diagonal matrix with blocks $\mathbf{A}_k = \beta_k \nu_k \mathbf{W}_k$ for each cluster and blocks $\mathbf{B}_n = -\mathbf{r}_n^T \mathbf{r}_n + \operatorname{diag}(\mathbf{r}_n)$ for each sample, where $\mathbf{r}_n$ is the nth row of the responsibility matrix $\mathbf{R}$ except for element r_{nK}, that is $\mathbf{r}_n = [r_{n1} \cdots r_{nK-1}]$. $\operatorname{diag}(\mathbf{a})$ is used here to denote a square matrix which has the elements of vector $\mathbf{a}$ on its diagonal.

The inverse of matrix $\mathbf{G}$ required for the evaluation of the natural gradient is easily calculated by inverting the individual blocks $\mathbf{A}_k$, $k = 1 \dots K$ and $\mathbf{B}_n$, $n = 1 \dots N$. We can now also motivate our earlier discussion about the number of degrees of freedom in responsibilities. Had we considered the number of degrees of freedom to be NK, the row vector $\mathbf{r}_n$ would have consisted of the whole nth row of matrix $\mathbf{R}$. This would have made the matrices $\mathbf{B}_n$ singular and we would not have been able to evaluate the natural gradient.

It was shown in [12] that the M-step of the VB EM algorithm can be regarded as natural gradient descent. Consequently, it is straightforward to show that the natural gradient update of the mean $\mathbf{m}_k$ equals the VB EM update equation (40) when a constant step size of $\lambda = 1$ is used. The main difference of the NCG and VB EM algorithms for learning the mixture of Gaussians is therefore the different way of updating the responsibilities complemented by the use of conjugate directions and line search in NCG.

V. Experiments

The priors are set to the following values for all the experiments that follow: $\alpha_0 = 1$, $\beta_0 = 1$, $\nu_0 = D$, $\mathbf{W}_0 = \frac{4}{D}\mathbf{I}$ and $\mathbf{m}_0 = \mathbf{0}$. These priors can be interpreted to describe our prior beliefs of the model when we anticipate having Gaussian components near the origin but are fairly uncertain about the number components.

The initial number of components is set to $K = 8$ unless otherwise mentioned with each component having a randomly generated initial mean $\mathbf{m}_k$ drawn from a Gaussian distribution with mean $\mathbf{m} = \mathbf{0}$ and covariance $\mathbf{\Sigma} = 0.16\mathbf{I}$. Other distribution parameters are initially set to the following values: $\alpha_k = 1$, $\beta_k = 10$, $\nu_k = D$ and $\mathbf{W}_k = \frac{4}{D}\mathbf{I}$ for all k.

All the experiments are repeated 30 times with different initial means because the performance of the algorithms can be greatly affected by the choice of initial values. The algorithms were considered to have converged when $\mathcal{C}_{k-1} - \mathcal{C}_k < \varepsilon$ for two consecutive iterations where $\varepsilon = 10^{-8}N$ unless otherwise mentioned.

A. Artificial Data

The artificial datasets used to compare the different algorithms are shown in Figure 3. The cluster data shown in Figure 3(a) is drawn from a mixture of 5 Gaussians. All the components are spherical and the center points of the components are $(0, 0)$ and $(\pm R, \pm R)$ where R can be changed. The data shown in Figure 3(a) has $R = 0.3$. All the components have mixing coefficient $\pi_k = 0.2$ and the number of data points is $N = 1000$ unless otherwise mentioned. The spiral data shown in Figure 3(b) is a three dimensional helix which is not drawn from a mixture of Gaussians. This dataset also has $N = 1000$ data points.

1) Cluster Data: When different gradient-based algorithms are compared using cluster data with $R = 0.3$, the results shown in Figure 4(a) are obtained. It can be seen that the standard gradient and CG algorithms have problems locating even a decent optimum. Using the natural gradient,

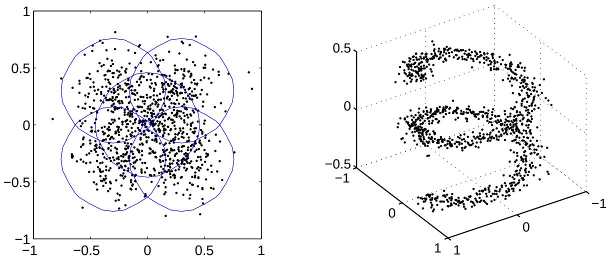


Fig. 3. Cluster data shown in Figure (a) consists of data drawn from a mixture of Gaussians with five identical components whose distance from each other can be changed. Spiral data shown in Figure (b) forms a three dimensional helix.

the quality of the optimum can be improved while NCG further improves the performance. It should be emphasized that the time scale of Figure 4(a) is logarithmic. Standard gradient is thus over 100 times slower than NCG. In contrast to other experiments discussed here, this experiment was conducted using the values $N = 500$, $\varepsilon = 10^{-7}N$ and the initial number of components $K = 5$ in order to make the standard gradient converge in a reasonable time. Out of these algorithms, only NCG is used in further experiments.

When the performance of NCG, VB EM and VB EM which is accelerated using pattern searches is compared using cluster data with $R = 0.3$, all the algorithms find the same solution. The computational performance of NCG and VB EM with pattern searches is quite similar while the performance of VB EM is slightly inferior to these two algorithms.

The same algorithms can also be compared with different values of R. Figure 4(d) shows the CPU time required for convergence of VB EM, VB EM with pattern search and NCG. For each value of R, the experiment is repeated 30 times with different initializations. The CPU time shown is the median of these experiments. It can be seen that with small values of R NCG outperforms VB EM while with large values of R VB EM performs better. Curiously, VB EM with pattern search seems to achieve good results with all values of R. All algorithms achieved approximately the same cost function values in this experiment. Thus, the CPU times shown can be compared directly.

2) Spiral Data: Using the spiral data, it can be seen both by looking at the learning curves of Figure 4(b) and the convergence results of Figure 4(e) that NCG is able to find much better optima than the other algorithms. It should be especially noted that none of the VB EM runs are able to locate the best optima found by NCG.

The results of this experiment are however greatly affected by the location and scaling of the dataset. If the helix is moved to the upper half space, the performance of NCG is further improved while scaling of the z-coordinate to the interval $[-1, 1]$ makes the algorithms mostly converge to the same optima.

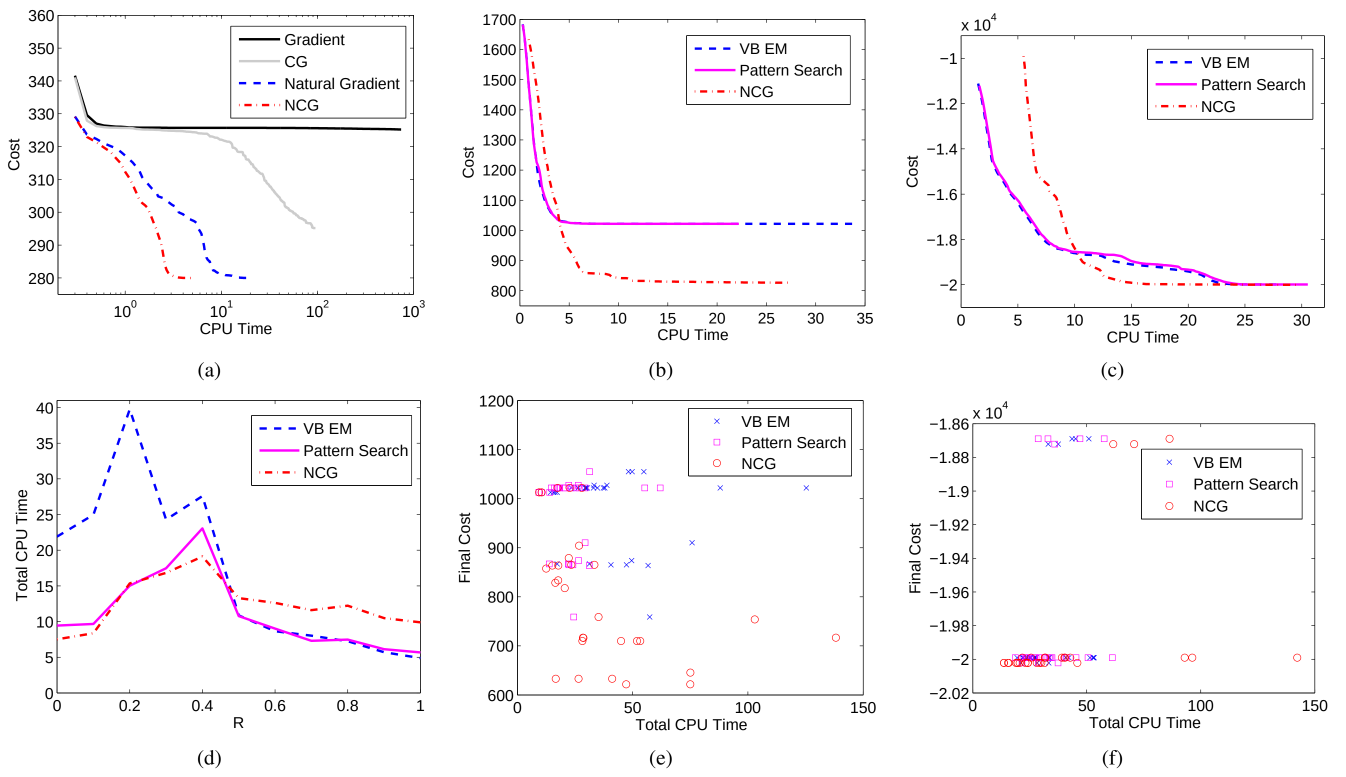


Fig. 4. *Left: Cluster data.* Figure (a) shows a comparison of gradient-based algorithms using cluster data with $R = 0.3$. Note that the time scale is logarithmic. Figure (d) shows the median total CPU time required for convergence as a function of R in cluster data. *Middle: Spiral data.* Figure (b) shows the median cost of 30 runs as a function of CPU time during learning. Figure (e) shows the final cost when the algorithms have converged and the corresponding CPU time for all the 30 initializations. *Right: Image data.* Figure (c) shows the median cost of 30 runs as a function of CPU time during learning of the image data. Figure (f) shows the final cost when the algorithms have converged and the corresponding CPU time for all the 30 initializations.

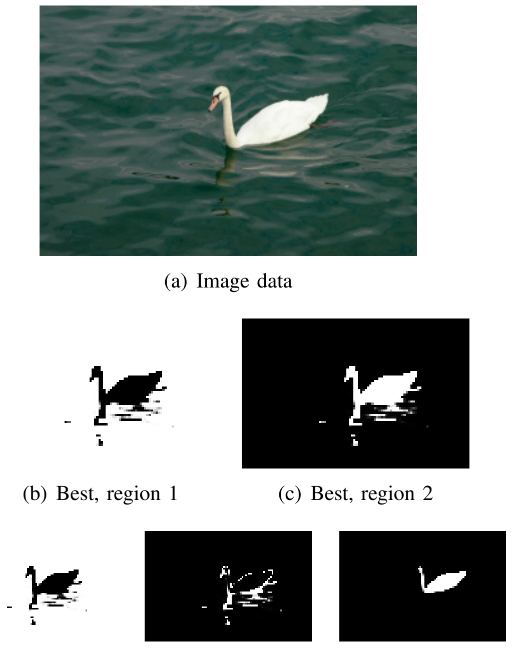
B. *Image Data*

In order to be able to compare the different algorithms using real world data, the algorithms were applied to the task of image segmentation. The goal is to divide a digital image into meaningful regions so that some characteristic of the image significantly changes on the boundary of two regions. This can be achieved for example by fitting a MoG to the image data which is interpreted so that each pixel of the image is a five dimensional data point with three color coordinates and two spatial coordinates. Segmented regions of the image can then be constructed using the responsibilities as measure of which region each pixel should be part of. It should be noted that in the variational approach the number of regions does not have to be determined beforehand but is instead selected automatically during learning. It should also be emphasized that there exists many other algorithms more suitable for image segmentation than the ones used here. Image segmentation is used here simply in order to easily obtain real world data where the results of the experiments are easily visualized. Figure 5(a) shows a 100×66 pixel image of a swan where the desired outcome of the segmentation is clearly the separation of the swan and the background. The image data is scaled so that it is within the five dimensional hypercube which has its center point in the origin of the space and a side length of 2.

Comparison of VB EM, VB EM with pattern search and NCG using the image data produces results shown on Figures 4(c) and 4(f). From the median learning curves of Figure 4(c), it can be seen that the initial convergence of NCG is faster than with the other algorithms but nevertheless no big differences in the convergence speed to the final optimum are noted. Examination of Figure 4(f) however reveals an interesting phenomenon. The best optimum is reached with 16 NCG runs out of a total of 30 while both in the case of VB EM and pattern search only 3 runs reached the best optimum. Although the difference in the cost function value of these optima is small, there is a big difference in the outcome of the segmentation task. This can be seen by plotting the responsibilities of each component in the final solution. This is done in Figure 5 where white represents responsibility 1 and black responsibility 0. It can be seen that the best optimum corresponds to the desired solution of segmentation into two regions while the second best optimum corresponds to a situation where the image is segmented into three regions. However, with some other images (not reported here), NCG found worse local minima than VB EM.

VI. Discussion

By looking at the experimental results of Section V, it can be said that NCG clearly outperforms other gradient-based algorithms. This is in line with previous results using


(a) Image data

(b) Best, region 1 (c) Best, region 2

(d) 2nd best, region 1 (e) 2nd best, region 2 (f) 2nd best, region 3

Fig. 5. Visual comparison of the best and the second best optimum found from the image data. It can be seen that the second best optimum corresponds to a situation where the image is divided into three regions instead of the desired two regions.

NCG [6]. It can also be said that NCG is highly competitive against algorithms based on VB EM. While in many cases the median performance of NCG is almost equal to algorithms based on VB EM, it seems to generally find the best optimum more often than the other algorithms. With the current implementation, it depends on the data which algorithm performs the best.

It might be possible to further improve the performance of NCG. The CPU time required by the algorithm is mainly spent on two things: on the evaluation of the gradient and on performing the line search. Therefore it would be worthwhile to study how various other line search methods compare against the quadratic polynomial interpolation based line search used here. It might also be possible to completely eliminate the need for a line search by using a scaled conjugate gradient algorithm [8] with natural gradient.

Further research is also required on determining exactly what kind of data is most suitable for the NCG algorithm. It is known that EM-like algorithms are prone to slow convergence in situations where the inference of latent variables is difficult, such as in the case of the cluster data with small values of R [11]. Based on the results shown in Figure 4(d), it can be said that NCG seems to be superior to VB EM in such cases. Interestingly, the pattern search method seems to also remarkably improve the performance of VB EM with such data.

VII. Conclusions

The aim of this work was to gain knowledge on the performance of the natural conjugate gradient algorithm on learning models in the conjugate-exponential family. When compared to the standard algorithm for performing variational inference in this family of models, the variational Bayesian expectation-maximization algorithm, we acquired experimental data which suggests that the NCG algorithm is highly competitive with VB EM when used to learn the mixture of Gaussians model. Especially the quality of the optima found using NCG seems to outperform VB EM in some cases. It should also be emphasized that while being a fairly complex algorithm to derive and implement, NCG generalizes to a broader family of models than VB EM.

On the question of which one of the compared algorithms is the best choice for learning the mixture of Gaussians model, it is fairly easy to conclude that given its simplicity and good performance across a wide range of data sets VB EM accelerated with pattern searches is the best choice. However, with certain types of data NCG might help finding a better local optimum.

Acknowledgements

This work was supported in part by the IST Program of the European Community, under the PASCAL2 Network of Excellence. AH was supported by the Academy of Finland (decision 121179). This publication only reflects the authors' views.

References

[1] S. Amari. *Differential-Geometrical Methods in Statistics*. Number 28 in Lecture Notes in Statistics. Springer-Verlag, 1985.

[2] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[3] C. M. Bishop. *Pattern recognition and machine learning*. Springer Science+Business Media, LLC., 2006.

[4] A. González and J. R. Dorronsoro. A note on conjugate natural gradient training of multilayer perceptrons. In *Proc. Int. Joint Conf. on Neural Networks, IJCNN 2006*, pages 887–891, 2006.

[5] T. Heskes. On "natural" learning and pruning in multilayered perceptrons. *Neural Computation*, 12(4):881–901, 2000.

[6] A. Honkela, M. Tornio, T. Raiko, and J. Karhunen. Natural conjugate gradient in variational inference. In *Proc. 14th Int. Conf. on Neural Information Processing (ICONIP 2007)*, volume 4985 of *LNCS*, pages 305–314, Kitakyushu, Japan, 2008. Springer-Verlag, Berlin.

[7] A. Honkela, H. Valpola, and J. Karhunen. Accelerating cyclic update algorithms for parameter estimation by pattern searches. *Neural Processing Letters*, 17(2):191–203, 2003.

[8] M. F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.

[9] M. K. Murray and J. W. Rice. *Differential Geometry and Statistics*. Chapman & Hall, 1993.

[10] E. Polak and G. Ribiére. Note sur la convergence de méthodes de directions conjugées. *Revue Française d'Informatique et de Recherche Opérationnelle*, 16:35–43, 1969.

[11] R. Salakhutdinov, S. Roweis, and Z. Ghahramani. Optimization with EM and expectation-conjugate-gradient. In *Proc. 20th International Conference on Machine Learning (ICML 2003)*, pages 672–679, 2003.

[12] M. Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681, 2001.

[13] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical Report CMU-CS-94-125, School of Computer Science, Carnegie Mellon University, 1994.

[14] H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 14(11):2647–2692, 2002.