Journal of Machine Learning Research 11 (2010) 3235-3268 Submitted 7/09, Revised 2/10; Published 11/10

Approximate Riemannian Conjugate Gradient Learning for Fixed-Form Variational Bayes

Antti Honkela[*] ANTTI.HONKELA@TKK.FI
Tapani Raiko[*] TAPANI.RAIKO@TKK.FI
Mikael Kuusela MIKAEL.KUUSELA@TKK.FI
Matti Tornio MATTI.TORNIO@TKK.FI
Juha Karhunen JUHA.KARHUNEN@TKK.FI
Department of Information and Computer Science
Aalto University School of Science and Technology
P.O. Box 15400
FI-00076 AALTO, Finland

Editor: Manfred Opper

Abstract

Variational Bayesian (VB) methods are typically only applied to models in the conjugate-exponential family using the variational Bayesian expectation maximisation (VB EM) algorithm or one of its variants. In this paper we present an efficient algorithm for applying VB to more general models. The method is based on specifying the functional form of the approximation, such as multivariate Gaussian. The parameters of the approximation are optimised using a conjugate gradient algorithm that utilises the Riemannian geometry of the space of the approximations. This leads to a very efficient algorithm for suitably structured approximations. It is shown empirically that the proposed method is comparable or superior in efficiency to the VB EM in a case where both are applicable. We also apply the algorithm to learning a nonlinear state-space model and a nonlinear factor analysis model for which the VB EM is not applicable. For these models, the proposed algorithm outperforms alternative gradient-based methods by a significant margin.

Keywords: variational inference, approximate Riemannian conjugate gradient, fixed-form approximation, Gaussian approximation

1. Introduction

Bayesian methods have recently gained popularity in machine learning. This is at least partially due to their robustness against overfitting compared to maximum likelihood and other methods based on point estimates. Variational Bayesian (VB) methods provide an efficient and often sufficiently accurate deterministic approximation to Bayesian learning (Bishop, 2006). Mean field type VB also has the benefit that its objective function can be used for choosing the model structure or model order. Most work on variational methods has focused on the class of conjugate exponential models for which simple EM-like learning algorithms can be derived easily (Ghahramani and Beal, 2001; Winn and Bishop, 2005). These models and algorithms are computationally convenient but they rule out many interesting model types.

∗. These authors contributed equally to this work. This project may be found at `http://www.cis.hut.fi/projects/bayes/`.

Many practically important models are not in the conjugate-exponential family and they have received far less attention in the VB literature. In this paper we present an efficient general method for applying VB learning to these more general models. The method could be used to speed up, for instance, the Gaussian variational approximation method of Opper and Archambeau (2009), or other previous more specific methods (e.g., Lappalainen and Honkela, 2000; Valpola and Karhunen, 2002).

Our method is based on first selecting the functional form of the approximation. For parts of the model that are conjugate-exponential, the corresponding factorised exponential family distribution is often a good choice, while in general we may use something else such as a multivariate Gaussian.

After fixing the functional form, we must be able to evaluate the variational free energy as a function of the variational parameters. The details of this step depend on the model—often most terms can be evaluated analytically while others may require computing some bounds (Jordan et al., 1999) or applying some linearisation techniques (see, e.g., Honkela and Valpola, 2005) or other approximations.

Once the free energy is known, we are left with a typically high-dimensional optimisation problem. Here[1] we propose using an approximate conjugate gradient algorithm that utilises the Riemannian geometry of the space of the approximations to speed up convergence. One of the main contributions of this paper is to use the geometry of the approximations. This is in contrast to more common applications of Riemannian geometry in natural gradient methods using the geometry of the predictive model. The geometry of the approximations is the natural choice if the variational inference is viewed as an optimisation problem in the space of approximating distributions. Furthermore, the geometry of the approximation is often much simpler, leading to more efficient computation and generic algorithms. The computational complexity of operations with the Fisher information matrix determining the geometry can be linear if the approximation is fully factorising or if its multivariate Gaussian blocks have a tree-like dependence structure, for instance. The resulting algorithm can provide dramatic speedups of potentially several orders of magnitude over state-of-the-art Euclidean conjugate gradient methods.

In previous machine learning algorithms Riemannian geometry is usually invoked through the natural gradient of Amari (1998). There, the aim has been to use maximum likelihood to directly update the model parameters $\boldsymbol{\theta}$ taking into account the geometry imposed by the predictive distribution of the data $p(\boldsymbol{X}|\boldsymbol{\theta})$. The resulting geometry is often quite complicated as the effects of different parameters cannot be separated and the Fisher information matrix is relatively dense. Recently Girolami and Calderhead (2011) have applied this in a Bayesian setting as a method to speed up Hamiltonian Monte Carlo samplers. In this paper, only the simpler geometry of the VB approximating distributions $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ is used. Because the approximations are often chosen to minimise dependencies between different parameters $\boldsymbol{\theta}$, the resulting Fisher information matrix with respect to the variational parameters $\boldsymbol{\xi}$ will be mostly diagonal and hence easy to work with.

The rest of the paper is organised as follows. Section 2 introduces the background in variational Bayes and information geometry. The proposed approximate Riemannian conjugate gradient learning algorithm is introduced in Section 3. The method is demonstrated in three case studies: a Gaussian mixture model, a nonlinear state-space model and a nonlinear factor analysis model in Secs. 4, 5 and 6, respectively. We end with discussion in Section 7.

1. This paper is an extended version of the earlier conference paper (Honkela et al., 2008).

2. Background

The approximate Riemannian conjugate gradient learning algorithm follows very naturally from an optimisation view of variational Bayes and the Riemannian geometry of probability distributions in information geometry. We will start with a brief introduction to both of these techniques separately.

2.1 Variational Bayes

Variational Bayesian (VB) learning (Jordan et al., 1999; Ghahramani and Beal, 2001; Bishop, 2006) is based on approximating the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{X})$ with a tractable approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$, where $\boldsymbol{X}$ is the data, $\boldsymbol{\theta}$ are the unobserved variables (including both the parameters of the model and the latent variables), and $\boldsymbol{\xi}$ are the variational parameters of the approximation (such as the mean and the variance of a Gaussian approximation). The approximation is fitted by minimising the free energy

$$\mathcal{F}(q(\boldsymbol{\theta}|\boldsymbol{\xi})) = E_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}\left\{\log \frac{q(\boldsymbol{\theta}|\boldsymbol{\xi})}{p(\boldsymbol{X},\boldsymbol{\theta})}\right\} = D_{\mathrm{KL}}(q(\boldsymbol{\theta}|\boldsymbol{\xi})\|p(\boldsymbol{\theta}|\boldsymbol{X})) - \log p(\boldsymbol{X}). \tag{1}$$

This is equivalent to minimising the Kullback-Leibler (KL) divergence $D_{\mathrm{KL}}(q\|p)$ between q and p (Bishop, 2006). The negative free energy also provides a lower bound on the marginal log-likelihood, that is, $\log p(\boldsymbol{X}) \geq -\mathcal{F}(q(\boldsymbol{\theta}|\boldsymbol{\xi}))$ due to non-negativity of the KL-divergence.

Typical classes of approximations used in VB include factorising approximations, most often starting from assuming the latent variables and parameters to be independent and extending to the fully factorising mean-field approximation, and approximations having a fixed functional form, such as a Gaussian.

In the former case, learning is typically performed using the variational Bayesian expectation maximisation (VB EM) algorithm which alternates between minimising the free energy with respect to the distribution of the latent variables (VB-E step) and the distribution of the parameters (VB-M step) (Bishop, 2006).

In the case of approximation with a fixed functional form, EM-like updates are usually not available and generic gradient-based optimisation methods have to be used (see, e.g., Opper and Archambeau, 2009). This is often very challenging in practice, as the problems are quite high dimensional and the lack of specific knowledge of interactions of the parameters that define the geometry of the problem can seriously hinder generic optimisation tools. Such methods have nevertheless been applied to a number of models that are not in the conjugate exponential family, such as multi-layer perceptron (MLP) networks (Barber and Bishop, 1998), kernel classifiers (Seeger, 2000), nonlinear factor analysis (Lappalainen and Honkela, 2000; Honkela and Valpola, 2005; Honkela et al., 2007) and nonlinear dynamical models (Valpola and Karhunen, 2002; Archambeau et al., 2008), non-conjugate variance models (Valpola et al., 2004; Raiko et al., 2007) as well as Gaussian process latent variable models (Titsias and Lawrence, 2010). Optimisation methods have included the conjugate gradient algorithm and heuristic speed-ups, but the use of a Riemannian conjugate gradient algorithm for VB as proposed in this paper is novel.

In practice, convergence of conjugate gradient algorithms in latent variable models is often really slow. To get an intuition of why this is the case, let us consider a generic latent variable model with latent variables that connect to one observation each and parameters that connect to a number of observations. As we increase the number of observations, the gradient with respect to the parameters grows linearly, whereas the gradient with respect to the latent variables stays constant.

As a result, with a reasonable number of observations, conjugate gradient algorithms are forced to take very small steps to avoid overshooting the parameters, and as a result the latent variables are hardly changed at all. The Riemannian gradient provides an automatic remedy to this problem by properly scaling the gradient.

2.2 Information Geometry and Optimisation on Riemannian Manifolds

When applying a generic optimisation algorithm to a problem such as optimising the free energy (1), a lot of background information on the geometry of the problem is lost. The parameters $\boldsymbol{\xi}$ of $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ can have different roles as location, shape, and scale parameters, and their effect is influenced by other parameters. This implies that the geometry of the problem is in most cases not Euclidean.

Riemannian geometry studies smooth manifolds $\{\mathcal{M}|\boldsymbol{\xi}\}$ that are locally diffeomorphic with $\mathbb{R}^n$. The manifold has an inner product $\langle\cdot,\cdot\rangle_k$ defined at every point $\boldsymbol{\xi}_k$ of the manifold. The inner product is defined for vectors in tangent space $T_{\boldsymbol{\xi}_k}$ of $\mathcal{M}$ at $\boldsymbol{\xi}_k$ as

$$\langle \mathbf{x}, \mathbf{y} \rangle_k = \mathbf{x}^T \mathbf{G}(\boldsymbol{\xi}_k) \mathbf{y} = \sum_{i,j} x_i y_j g_{ij}(\boldsymbol{\xi}_k), \tag{2}$$

where $\mathbf{G}(\boldsymbol{\xi}_k)$ is the Riemannian metric tensor at $\boldsymbol{\xi}_k$. Most Riemannian manifolds are curved, with geodesics as the counterparts of Euclidean straight lines as length-minimising curves between two points (Murray and Rice, 1993).

Information geometry studies the Riemannian geometric structure of the manifold of probability distributions (Amari, 1985). It has been applied to derive efficient natural gradient learning rules for maximum likelihood algorithms in independent component analysis (ICA) and multi-layer perceptron (MLP) networks (Amari, 1998). The approach has been used in several other problems as well, for example in analysing the properties of an on-line variational Bayesian EM method (Sato, 2001).

For probability distributions, the most natural metric is given by the Fisher information

$$g_{ij}(\boldsymbol{\xi}) = I_{ij}(\boldsymbol{\xi}) = E\left\{ \frac{\partial \log q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_i} \frac{\partial \log q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_j} \right\} = E\left\{ -\frac{\partial^2 \log q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_i \partial \xi_j} \right\}. \tag{3}$$

Here the last equality is valid if $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ is twice continuously differentiable (Murray and Rice, 1993). Fisher information is a unique metric for probability distributions in the sense that it is the only metric which is invariant with respect to transformations to sufficient statistics (Amari and Nagaoka, 2000).

In a Riemannian space, the direction of steepest ascent of a function $\mathcal{F}(\boldsymbol{\xi})$ is given by the Riemannian or natural gradient

$$\tilde{\nabla}\mathcal{F}(\boldsymbol{\xi}) = \mathbf{G}^{-1}(\boldsymbol{\xi}) \nabla \mathcal{F}(\boldsymbol{\xi}) \tag{4}$$

instead of the regular gradient $\nabla\mathcal{F}(\boldsymbol{\xi})$. This can be verified by finding the maximum of $\mathcal{F}(\boldsymbol{\xi}+\Delta\boldsymbol{\xi})$ subject to the constraint $||\Delta\boldsymbol{\xi}||^2_{\boldsymbol{\xi}} = \langle \Delta\boldsymbol{\xi}, \Delta\boldsymbol{\xi} \rangle_{\boldsymbol{\xi}} = \Delta\boldsymbol{\xi}^T \mathbf{G}(\boldsymbol{\xi}) \Delta\boldsymbol{\xi} < \varepsilon^2$. The relation between gradient and Riemannian gradient is illustrated in Figure 1.

This choice of geometry between Euclidean and Riemannian is, however, independent of the choice of the optimisation algorithm, and recently several authors have combined conjugate gradient methods with the Riemannian or natural gradient (Smith, 1993; González and Dorronsoro, 2008; Honkela et al., 2008). In principle this can be achieved by replacing all vector space operations in

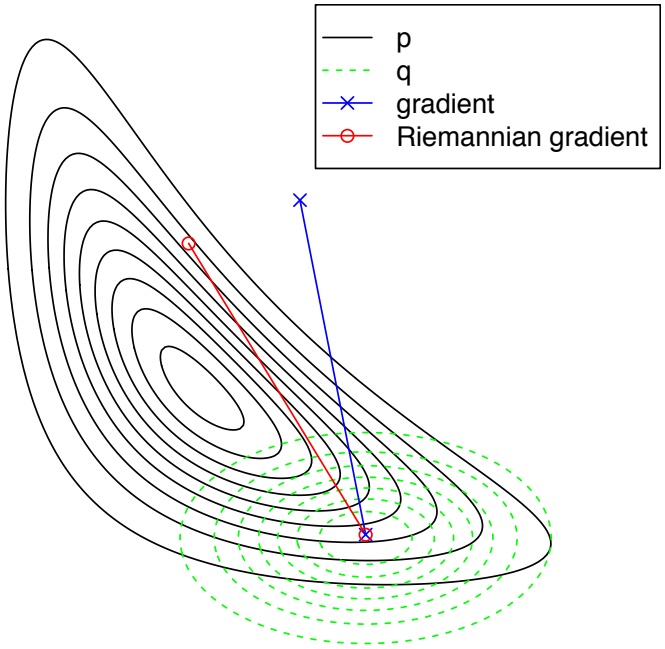

Figure 1: Gradient and Riemannian gradient directions are shown for the mean of distribution q. VB learning with a diagonal covariance is applied to the posterior $p(x,y) \propto \exp[-9(xy-1)^2 - x^2 - y^2]$. The Riemannian gradient strengthens the updates in the directions where the uncertainty is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

the conjugate gradient algorithm with their Riemannian counterparts: Riemannian inner products and norms, parallel transport of gradient vectors between different tangent spaces as well as line searches and steps along geodesics in the Riemannian space. In practical algorithms some of these can be approximated by their flat-space counterparts. We shall apply the approximate Riemannian conjugate gradient (RCG) method which implements Riemannian (natural) gradients, inner products and norms but uses flat-space approximations of the others as our optimisation algorithm of choice throughout the paper. As shown in Appendix A, these approximations do not affect the asymptotic convergence properties of the algorithm. The difference between gradient and conjugate gradient methods is illustrated in Figure 2.

In this paper we propose using the Riemannian structure of the distributions $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ to derive more efficient algorithms for approximate inference and especially VB using approximations with a fixed functional form. This differs from the traditional natural gradient learning by Amari (1998) which uses the Riemannian structure of the predictive distribution $p(\boldsymbol{X}|\boldsymbol{\theta})$. The proposed method can be used to jointly optimise all the parameters $\boldsymbol{\xi}$ of the approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$, or in conjunction with VB EM for some parameters.

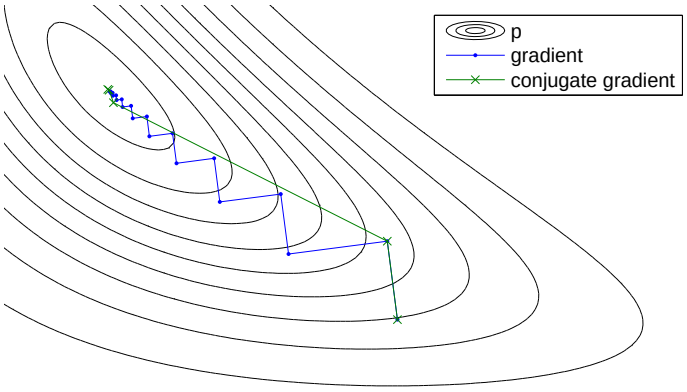

Figure 2: Gradient and conjugate gradient updates are applied to finding the maximum of the posterior $p(x,y) \propto \exp[-9(xy-1)^2 - x^2 - y^2]$. The step sizes that maximise p are used. Note that the first steps are the same, but following gradient updates are orthogonal whereas conjugate gradient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

2.3 Information Geometry of VB EM

The optimal VB approximation has an information-geometric interpretation as a specific projection of the true posterior to a manifold of tractable distributions (Tanaka, 2001). This interpretation is equally valid for all optimisation methods.

Amari (1995) has also presented the geometric interpretation of the EM algorithm as alternating projections for E- and M-steps. This asymmetric view does not directly generalise to the VB EM method when used to infer distributions over all parameters, because VB EM is symmetric with respect to different parameters.

By embedding the VB-E step update within the VB-M step with point estimates and considering the resulting update, the VB EM algorithm for conjugate exponential family models can be interpreted as a natural gradient method (Sato, 2001). It therefore implicitly optimally utilises the Riemannian geometric structure of $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ (Amari, 1998). Nevertheless, the EM-based methods are prone to slow convergence, especially under low noise, even though more elaborate optimisation schemes can speed up their convergence somewhat. It is worth pointing out that this correspondence of VB EM is with the regular natural gradient algorithm, not Riemannian (natural) conjugate gradients as proposed in this paper.

3. Approximate Riemannian Conjugate Gradient Learning for Fixed-Form VB

Given a fixed-form approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ and the free energy $\mathcal{F}(q(\boldsymbol{\theta}|\boldsymbol{\xi}))$, it is possible to use standard gradient-based optimisation techniques to minimise the free energy with respect to $\boldsymbol{\xi}$. We will use VB EM updates for some variational parameters $\boldsymbol{\xi}^{\mathrm{EM}}$ and RCG for others $\boldsymbol{\xi}^{\mathrm{RCG}}$.

Instead of a regular Euclidean gradient algorithm, we optimise the free energy using a conjugate gradient algorithm that is adapted to Riemannian space by using Riemannian inner products and norms instead of Euclidean ones. Steps are still taken along Euclidean straight lines and the step

length is determined using a line search. We call this the (approximate) Riemannian conjugate gradient (RCG) algorithm.

Our RCG is an approximation of a true Riemannian conjugate gradient algorithm (Smith, 1993), in which the steps are taken along geodesic curves and tangent vectors evaluated at different points are transformed to the same tangent space using parallel transport along a geodesic. For small step sizes and geometries which are locally close to Euclidean, the approximations that we have made still retain many of the benefits of the exact algorithm while greatly simplifying the computations. Edelman et al. (1998) showed that near the solution Riemannian conjugate gradient method differs from the flat space version of conjugate gradient only by third order terms, and therefore both algorithms converge quadratically near the optimum. This convergence property is demonstrated in detail for our approximation in Appendix A.

The search direction for the RCG method is given by

$$\mathbf{p}_k = -\tilde{\mathbf{g}}_k + \beta \mathbf{p}_{k-1},$$

where $\tilde{\mathbf{g}}_k = \tilde{\nabla}\mathcal{F}(\boldsymbol{\xi})$ is the Riemannian gradient of Equation (4). The coefficient β is evaluated using the Polak-Ribiére formula (Nocedal, 1992; Smith, 1993; Edelman et al., 1998)

$$\beta = \frac{\langle \tilde{\mathbf{g}}_k, \tilde{\mathbf{g}}_k - \tilde{\mathbf{g}}_{k-1} \rangle_k}{||\tilde{\mathbf{g}}_{k-1}||^2_{k-1}}, \tag{5}$$

where $||\tilde{\mathbf{g}}_k||^2_k = \langle \tilde{\mathbf{g}}_k, \tilde{\mathbf{g}}_k \rangle_k$ is the squared Riemannian norm of $\tilde{\mathbf{g}}_k$ in the tangent space where $\tilde{\mathbf{g}}_k$ is defined, and $\langle \mathbf{x}, \mathbf{y} \rangle_k$ denotes the Riemannian inner product of Equation (2) in the same tangent space.

We also apply a Riemannian version of the Powell-Beale restart method (Powell, 1977): the search direction is reset to the negative gradient direction if

$$|\langle \tilde{\mathbf{g}}_{k-1}, \tilde{\mathbf{g}}_k \rangle_k| \geq 0.2 ||\tilde{\mathbf{g}}_k||^2_k. \tag{6}$$

Compared to the traditional conjugate gradient, the Equations (5) and (6) are similar with just the dot products of the vectors replaced with Riemannian inner products.

Once the search direction is determined, we use standard line search to find the final update. Because the evaluation of the objective function is computationally costly, it is worthwhile to consider line search methods that stop earlier rather than wasting many function evaluations on fine tuning the parameters.

An example implementation of the algorithm is summarised in Algorithm 1. The inputs include the probabilistic model p, the form of used posterior approximation q, the initialisations for the variational parameters $\boldsymbol{\xi}$, and the data set $\boldsymbol{X}$ which is implicitly used in the objective function $\mathcal{F}$. The algorithm returns the variational parameters $\boldsymbol{\xi}$ that solve the learning and inference problem of $q(\boldsymbol{\theta}|\boldsymbol{\xi})$.

3.1 Computational Considerations

The RCG method is efficient as the geometry is defined by the approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ and not the full model $p(\boldsymbol{X}|\boldsymbol{\theta})$ as in typical natural gradient methods. If the approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ is chosen such that disjoint groups of variables are independent, that is,

$$q(\boldsymbol{\theta}|\boldsymbol{\xi}) = \prod_i q_i(\boldsymbol{\theta}_i|\boldsymbol{\xi}_i),$$

Algorithm 1 An outline of an example Riemannian conjugate gradient algorithm for fixed-form VB. The presented method of integrating VB EM updates is only one of many possible alternatives.

function VB-RCG($p, q, \boldsymbol{\xi}_0 = (\boldsymbol{\xi}_0^{\mathrm{EM}}, \boldsymbol{\xi}_0^{\mathrm{RCG}}), \boldsymbol{X}$)
  $\mathbf{p}_0 = \mathbf{0}, \tilde{\mathbf{g}}_0 = \mathbf{1}$
  for $k = 1, 2, \ldots$ **do** ▷ Repeat until convergence
    for $\boldsymbol{\xi}^{(i)} \in \boldsymbol{\xi}^{\mathrm{EM}} = (\boldsymbol{\xi}^{(1)}, \ldots, \boldsymbol{\xi}^{(N)})$ **do**
      $\boldsymbol{\xi}_k^{(i)} \leftarrow \arg\min_{\boldsymbol{\xi}^{(i)}} \mathcal{F}(q(\boldsymbol{\theta}|\boldsymbol{\xi}_k^{(1)}, \ldots, \boldsymbol{\xi}_k^{(i-1)}, \boldsymbol{\xi}^{(i)}, \boldsymbol{\xi}_{k-1}^{(i+1)}, \ldots, \boldsymbol{\xi}_{k-1}^{(N)}, \boldsymbol{\xi}_{k-1}^{\mathrm{RCG}}))$
      ▷ VB EM for some parameters
    $\tilde{\mathbf{g}}_k \leftarrow \mathbf{G}^{-1}(\boldsymbol{\xi}_{k-1}^{\mathrm{RCG}}) \nabla_{\boldsymbol{\xi}_{k-1}^{\mathrm{RCG}}} \mathcal{F}(q(\boldsymbol{\theta}|\boldsymbol{\xi}_k^{\mathrm{EM}}, \boldsymbol{\xi}_{k-1}^{\mathrm{RCG}}))$ ▷ Riemannian gradient
    $\beta \leftarrow \frac{\langle \tilde{\mathbf{g}}_k, \tilde{\mathbf{g}}_k - \tilde{\mathbf{g}}_{k-1} \rangle_k}{\|\tilde{\mathbf{g}}_{k-1}\|_{k-1}^2}$ ▷ Polak-Ribiére formula
    $\mathbf{p}_k \leftarrow -\tilde{\mathbf{g}}_k + \beta \mathbf{p}_{k-1}$ ▷ Update direction
    $\alpha \leftarrow \arg\min_\alpha \mathcal{F}(q(\boldsymbol{\theta}|\boldsymbol{\xi}_k^{\mathrm{EM}}, \boldsymbol{\xi}_{k-1}^{\mathrm{RCG}} + \alpha \mathbf{p}_k))$ ▷ Line search
    $\boldsymbol{\xi}_k^{\mathrm{RCG}} \leftarrow \boldsymbol{\xi}_{k-1}^{\mathrm{RCG}} + \alpha \mathbf{p}_k$

the computation of the Riemannian gradient is simplified as the Fisher information matrix becomes block-diagonal. The required matrix operations can be performed very efficiently because

$$\mathrm{diag}(\mathbf{A}_1, \ldots, \mathbf{A}_n)^{-1} = \mathrm{diag}(\mathbf{A}_1^{-1}, \ldots, \mathbf{A}_n^{-1}).$$

The dimensionality of the problem space is often so high that working with the full matrix would not be feasible.

All vector operations needed in the RCG algorithm are of the form

$$\langle \tilde{\mathbf{g}}_k, \tilde{\mathbf{g}}_l \rangle_m = \langle \mathbf{G}_k^{-1} \mathbf{g}_k, \mathbf{G}_l^{-1} \mathbf{g}_l \rangle_m = \mathbf{g}_k^T \mathbf{G}_k^{-T} \mathbf{G}_m \mathbf{G}_l^{-1} \mathbf{g}_l \tag{7}$$

for some iterate indices k, l, m. This is further simplified in case $m = k$ where $\mathbf{G}_k^{-T} \mathbf{G}_m = \mathbf{I}$ and in case $m = l$ where $\mathbf{G}_m \mathbf{G}_l^{-1} = \mathbf{I}$. Depending on the structure of the Fisher information matrix, the operations can be performed as a series of solving linear systems and matrix products to exploit the sparsity. A practical example of this in the case of a Gaussian approximation is presented in Section 3.2.1.

Finally, it is worth to note that when updating only a subset of variational parameters $\boldsymbol{\xi}$ at a time, many terms in $\mathcal{F}$ are constant and can be disregarded when finding a minimum.

3.2 Gaussian Approximation

Most obvious applications of the Riemannian gradient method are with a Gaussian approximation. In that case, it is most convenient to use a simple fixed-point update rule for the covariance and a Riemannian conjugate gradient update only for the mean.

Let us consider the optimisation of the free energy (1) when the approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ is a multivariate Gaussian. The free energy can be decomposed as

$$\mathcal{F}(q(\boldsymbol{\theta}|\boldsymbol{\xi})) = E_{q(\boldsymbol{\theta}|\boldsymbol{\xi})} \left\{ \log \frac{q(\boldsymbol{\theta}|\boldsymbol{\xi})}{p(\boldsymbol{X}, \boldsymbol{\theta})} \right\} = E_{q(\boldsymbol{\theta}|\boldsymbol{\xi})} \left\{ \log q(\boldsymbol{\theta}|\boldsymbol{\xi}) \right\} + E_{q(\boldsymbol{\theta}|\boldsymbol{\xi})} \left\{ -\log p(\boldsymbol{X}, \boldsymbol{\theta}) \right\}.$$

The former term is the negative entropy of the approximation, which in the case of a multivariate Gaussian $q(\boldsymbol{\theta}|\boldsymbol{\mu},\boldsymbol{\Sigma})$ with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is

$$E_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}\left\{\log q(\boldsymbol{\theta}|\boldsymbol{\xi})\right\} = -\frac{1}{2}\log\det(2\pi e\boldsymbol{\Sigma}).$$

Straightforward differentiation yields a fixed point update rule for the covariance (Lappalainen and Miskin, 2000; Opper and Archambeau, 2009):

$$\boldsymbol{\Sigma}^{-1} = -2\nabla_{\boldsymbol{\Sigma}} E_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}\left\{\log p(\boldsymbol{X},\boldsymbol{\theta})\right\}, \tag{8}$$

where $\nabla_{\boldsymbol{\Sigma}}$ denotes the gradient with respect to $\boldsymbol{\Sigma}$. If the covariance matrix is assumed (block) diagonal, the same update rule applies for the (block) diagonal terms.

3.2.1 COMPUTING THE RIEMANNIAN METRIC TENSOR

For the univariate Gaussian distribution parametrised by mean and variance $q(\theta|\mu,v) = \mathcal{N}(\theta|\mu,v)$, we have

$$\log q(\theta|\mu,v) = -\frac{1}{2v}(\theta-\mu)^2 - \frac{1}{2}\log(v) - \frac{1}{2}\log(2\pi).$$

Furthermore,

$$E\left\{-\frac{\partial^2 \log q(\theta|\mu,v)}{\partial\mu^2}\right\} = \frac{1}{v}, \tag{9}$$

$$E\left\{-\frac{\partial^2 \log q(\theta|\mu,v)}{\partial v\partial\mu}\right\} = 0, \text{ and} \tag{10}$$

$$E\left\{-\frac{\partial^2 \log q(\theta|\mu,v)}{\partial v^2}\right\} = \frac{1}{2v^2}. \tag{11}$$

The vanishing of the cross term between the mean and the variance further supports using the simpler fixed point rule (8) to update the variances.

(a) (b) (c) (d)

Figure 3: The absolute change in the mean of the Gaussian in figures (a) and (b) and the absolute change in the variance of the Gaussian in figures (c) and (d) is the same. However, the relative effect is much larger when the variance is small as in figures (a) and (c) compared to the case when the variance is high as in figures (b) and (d) (Valpola, 2000).

In the case of univariate Gaussian distribution, the Riemannian gradient has a rather straightforward intuitive interpretation, which is illustrated in Figure 3. Compared to conventional gradient, Riemannian gradient compensates for the fact that changing the parameters of a Gaussian with small variance has much more pronounced effects than when the variance is large.

In case of multivariate Gaussian distribution parametrised by mean and covariance $q(\boldsymbol{\theta}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu},\boldsymbol{\Sigma})$, the elements of the Fisher information matrix corresponding to the mean are simply

$$E\left\{-\frac{\partial^2 \log q(\boldsymbol{\theta}|\boldsymbol{\mu},\boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}^T \partial \boldsymbol{\mu}}\right\} = \boldsymbol{\Sigma}^{-1}. \tag{12}$$

The Fisher information matrix is thus equal to the precision matrix $\mathbf{G}_k = \boldsymbol{\Lambda}_k = \boldsymbol{\Sigma}_k^{-1}$.

Typically the covariance matrix $\boldsymbol{\Sigma}$ is assumed to have a simple structure that makes working with it very efficient. Possible structures for a covariance matrix include full, diagonal, block diagonal, a Gaussian Markov random field with a specific structure, and a factor analysis covariance $\boldsymbol{\Sigma} = \mathbf{D} + \sum_{i=1}^{k} \mathbf{v}\mathbf{v}^T$, where $\mathbf{D}$ is a diagonal matrix, or $\boldsymbol{\Sigma}^{-1} = \mathbf{K}^{-1} + \mathrm{diag}(\mathbf{v})$ with a fixed $\mathbf{K}$ and only N parameters in $\mathbf{v}$ for an N-variate Gaussian (Opper and Archambeau, 2009). It is also possible to derive the geometry for the covariance of a multivariate Gaussian. The result does, however, depend on the specific structure of the covariance.

Assuming a structured Gaussian Markov random field approximation with a tree or blocked tree structure, the precision matrix will be sparse with a simple structure. This allows efficient computation of the operations needed in Equation (7). $\boldsymbol{\Lambda}_l^{-1}\mathbf{g}_l$ (and correspondingly for k) can be computed by solving the linear system $\boldsymbol{\Lambda}_l \mathbf{x} = \mathbf{g}_l$, which can be done in $O(N)$ time for N variables using a propagation algorithm in the tree. For a blocked tree formed of N/n blocks of size n, the complexity is $O(n^2 N)$. Examples of such algorithms for chains are given in Golub and Loan (1996), but a general tree can be handled analogously. The complexity of multiplication by $\boldsymbol{\Lambda}$ is similar.

Examples of Gaussian Markov random fields with this structure can be easily found in time series models, where the approximation for the state sequence $\boldsymbol{S} = (\mathbf{s}(1), \ldots, \mathbf{s}(T))$ is typically either a single "blocked" chain

$$q(\boldsymbol{S}) = \prod_{t=2}^{T} q(\mathbf{s}(t)|\mathbf{s}(t-1)) q(\mathbf{s}(1))$$

or a product of independent chains

$$q(\boldsymbol{S}) = \prod_i \left[q(s_i(1)) \prod_{t=2}^{T} q(s_i(t)|s_i(t-1)) \right]. \tag{13}$$

In a d dimensional model of length T, the time complexity of the Riemannian vector operations in RCG is $O(d^3 T)$ for the former and $O(dT)$ for the latter.

4. Case Study: Mixture of Gaussians

As the first case study, we consider the mixture-of-Gaussians (MoG) model as was done by Kuusela et al. (2009). In this case, we applied the RCG also for variables with a non-Gaussian approximation. Furthermore, the conjugate-exponential nature of the MoG model allows direct comparison with VB EM.

4.1 The Mixture-of-Gaussians Model

We consider a finite mixture of K Gaussians (Attias, 2000; Bishop, 2006)

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

The model	$p(\mathbf{X}\|\mathbf{Z},\boldsymbol{\mu},\boldsymbol{\Lambda}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\mathcal{N}(\mathbf{x}_n\|\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k^{-1})^{z_{nk}}$ $p(\mathbf{Z}\|\boldsymbol{\pi}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\pi_k^{z_{nk}}$
Key variables	$\mathbf{Z} = (z_{nk}), \boldsymbol{\theta} = (\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k, \boldsymbol{\pi})$
The approximation	$q(\mathbf{Z},\boldsymbol{\theta}) = q(\mathbf{Z})q(\boldsymbol{\theta})$
The update algorithm	Joint RCG updates for $\mathbf{Z}$ and means of $\boldsymbol{\mu}_k$, fixed point updates for variances of $\boldsymbol{\mu}_k$, VB EM updates for the rest

Table 1: Summary of the mixture-of-Gaussians model

where $\mathbf{x}$ is a D-dimensional random vector, $\boldsymbol{\pi} = [\pi_1 \cdots \pi_K]^T$ are the mixing coefficients, and $\boldsymbol{\mu}_k$ and $\boldsymbol{\Lambda}_k$ are the mean vector and the precision matrix of the kth Gaussian component, respectively.

In the case of the MoG model, the binary latent variables $\mathbf{Z}$ denote which one of the K Gaussian components has generated a particular observation $\mathbf{x}_n$ with $z_{nk} = 1$ denoting the component responsible for generating the observed data point $\mathbf{x}_n$. Let N denote the total number of observed data points.

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

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

The mixing coefficients have a conjugate Dirichlet prior

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

where $\boldsymbol{\alpha}_0 = [\alpha_0, \ldots, \alpha_0]^T$.

Similarly, the likelihood can be written as

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

In this case, the conjugate prior for the component parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}$ is given by the Gaussian-Wishart distribution

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

where the Wishart distribution is defined up to a normalising constant by the equation

$$\mathcal{W}(\boldsymbol{\Lambda}|\mathbf{W},\nu) \propto |\boldsymbol{\Lambda}|^{(\nu-D-1)/2}\exp\left(-\frac{1}{2}\mathrm{Tr}(\mathbf{W}^{-1}\boldsymbol{\Lambda})\right).$$

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

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

This resulting model can be illustrated with the graphical model shown in Figure 4.

We now make the factorising approximation

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

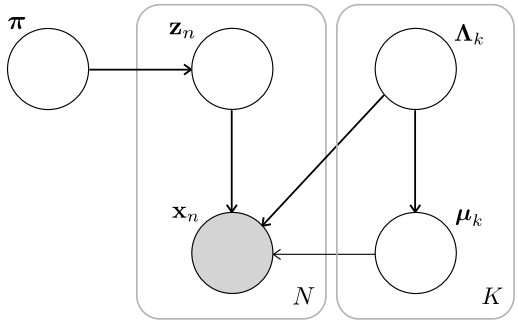

Figure 4: A graphical model representing the MoG model (Attias, 2000; Bishop, 2006), where the hyperparameters have been omitted for clarity. The observed data $\mathbf{X}$ are marked with a shaded circle. The rectangular plates denote the repetition of N observations $\mathbf{x}_n$ along with corresponding latent variables $\mathbf{z}_n$, and of the parameters of K mixture components.

which leads to an update rule for $q(\mathbf{Z})$ (VB-E step) and subsequently an update rule for $q(\boldsymbol{\pi},\boldsymbol{\mu},\mathbf{\Lambda})$ (VB-M step). The resulting approximate posterior distributions are

$$q(\mathbf{Z}) = \prod_{n=1}^{N}\prod_{k=1}^{K} r_{nk}^{z_{nk}} \tag{14}$$

and

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

where

$$q(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \tag{15}$$

and

$$q(\boldsymbol{\mu}_k,\mathbf{\Lambda}_k) = \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_k,(\beta_k\mathbf{\Lambda}_k)^{-1})\,\mathcal{W}(\mathbf{\Lambda}_k|\mathbf{W}_k,\nu_k). \tag{16}$$

The derivations of the VB EM and RCG learning algorithms for the MoG model are presented in Appendix B.

4.2 Experiments

The hyperparameters are set to the following values for all the experiments: $\alpha_0 = 1$, $\beta_0 = 1$, $\nu_0 = D$, $\mathbf{W}_0 = \frac{4}{D}\mathbf{I}$ and $\mathbf{m}_0 = \mathbf{0}$. These values 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 of the components.

The maximum 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 zero mean and covariance $0.16\mathbf{I}$. 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. The Powell-Beale restart scheme of Equation (6) is not used. Instead, the search direction is reset to the negative gradient direction

after $\sqrt{n}$ iterations, where n is the number of parameters updated with the gradient method. The optimisation is assumed to have converged when the improvement in free energy $|\mathcal{F}^t - \mathcal{F}^{t-1}| < \epsilon$ for two consecutive iterations with ϵ being separately specified for each of the experiments below.

It should be noted that this convergence criterion favours methods such as VB EM which typically takes more of cheaper and smaller steps, while the Riemannian gradient algorithm takes fewer larger steps that are computationally more demanding and take longer to reach the preset improvement threshold.

Because different initial means can produce significantly different results in terms of the required CPU time and achieved final free energy, all the experiments are repeated 30 times with different initialisations.

The artificial data set used to compare the different algorithms in learning the MoG model was drawn from a mixture of 5 spherical two-dimensional Gaussians with equal weights. The mean of the first component is at the origin while the means of the others are $(\pm R, \pm R)$. The constant $R = 0.3$ unless otherwise mentioned and the covariance of all the components is $0.03\mathbf{I}$.

When different gradient-based algorithms are compared using the artificial data containing $N = 500$ data points, the results shown in Figure 5 are obtained. It can be seen that the standard gradient descent and conjugate gradient (CG) algorithms have problems locating even a decent optimum within a reasonable time. Clearly the convergence criterion, which was set to $\epsilon = 10^{-7}N$, is too lax for these algorithms as the simulations are terminated before convergence to a good solution. Using the Riemannian gradient (RG) and further RCG radically improves the performance. Based on the curves, even the standard CG algorithm is more than 10 times slower than RCG. This experiment was conducted using a fairly small number of observations, a lax convergence criterion and the maximum number of components $K = 5$ in order to allow the standard gradient to finish in a reasonable time.

We also considered the L-BFGS algorithm (Byrd et al., 1995; Carbonetto, 2007) as a higher order Euclidean algorithm. L-BFGS is a limited-memory version of the popular quasi-Newton BFGS algorithm. It can be seen as a compromise between the fast converging but memory-intensive quasi-Newton methods and the less efficient conjugate gradient methods better suited for medium- to large-scale problems. The degree of this compromise is controlled by the memory length parameter m which was set to $m = 20$ in Figure 5. It was found out that, regardless of the value of m, the performance of L-BFGS was very similar to CG with the small deviations explained by the differences in the line search methods employed by the algorithms. It also turned out that the line search subroutine of the L-BFGS implementation had a tendency to converge prematurely to a poor solution. In order to circumvent this, the convergence criterion of the L-BFGS had to be tightened by a factor of 10^{-9} compared to the gradient-based algorithms.

We next compare RCG, RG and VB EM using different values of R. The number of observations was increased to $N = 1000$ and the convergence criterion was set to $\epsilon = 10^{-12}N$ in order to maximise the quality of the optima found. Figure 6 shows the median CPU time required for convergence for the different algorithms in 30 repeated experiments. It can be seen that with small values of R, RCG outperforms VB EM, while with large values of R, VB EM performs slightly better. This means that, at least in terms of this experiment, RCG performs better than VB EM when the latent variables are difficult to infer from the data.

Given the discussion of Section 2.3, it is not surprising to note that both VB EM and RG perform qualitatively in a fairly similar manner in the experiment of Figure 6. It should especially be noted that both methods suffer from significant slowdowns near the values of $R = 0.2$ and $R = 0.325$. On

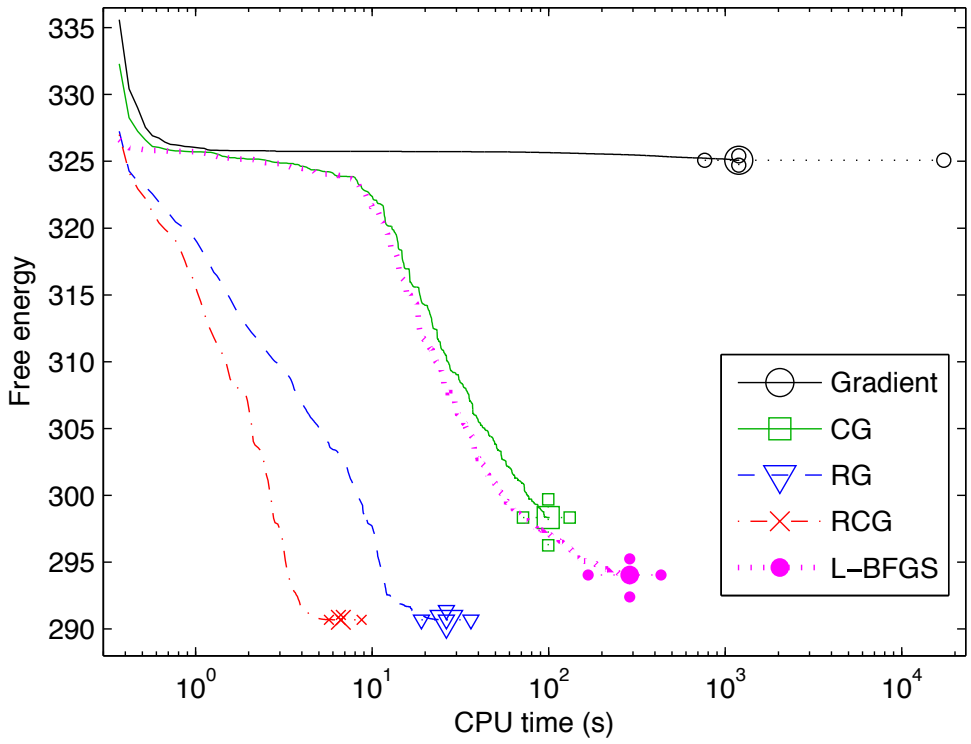

Figure 5: Convergence curves of gradient-based algorithms using the MoG model for artificial data with $R = 0.3$. The algorithms compared are the standard gradient descent, the conjugate gradient (CG), the Riemannian gradient (RG) and the Riemannian conjugate gradient (RCG) methods as well as the limited-memory BFGS (L-BFGS) algorithm. The curves shown are medians of 30 simulations drawn up to the median termination time. The smaller marks denote 25% and 75% quantiles of the termination time in the horizontal direction and the corresponding quantiles of the free energy at the median termination time in the vertical direction. Note that the time scale is logarithmic.

the other hand, the use of conjugate directions in the Riemannian space seems to result in a fairly uniform performance across all values of R.

There is some variation in the quality of the optima the different methods converge to. This is illustrated in Figure 7 for RCG and VB EM. There is no evidence for either algorithm consistently producing better results than the other. The only outlier in this data is with $R = 0.225$ where VB EM is the only algorithm to discover the global optimum. This is by no means typical, and with other data sets we have seen RCG sometimes consistently finding better optima. Based on the figure, RCG seems slightly more sensitive to local optima, but the result is not qualitatively different from VB EM which, to some extent, also suffers from the same problem.

Although the time complexity of one step of each of the compared algorithms is linear in the number of samples, the algorithms nevertheless perform differently as the number of samples in-

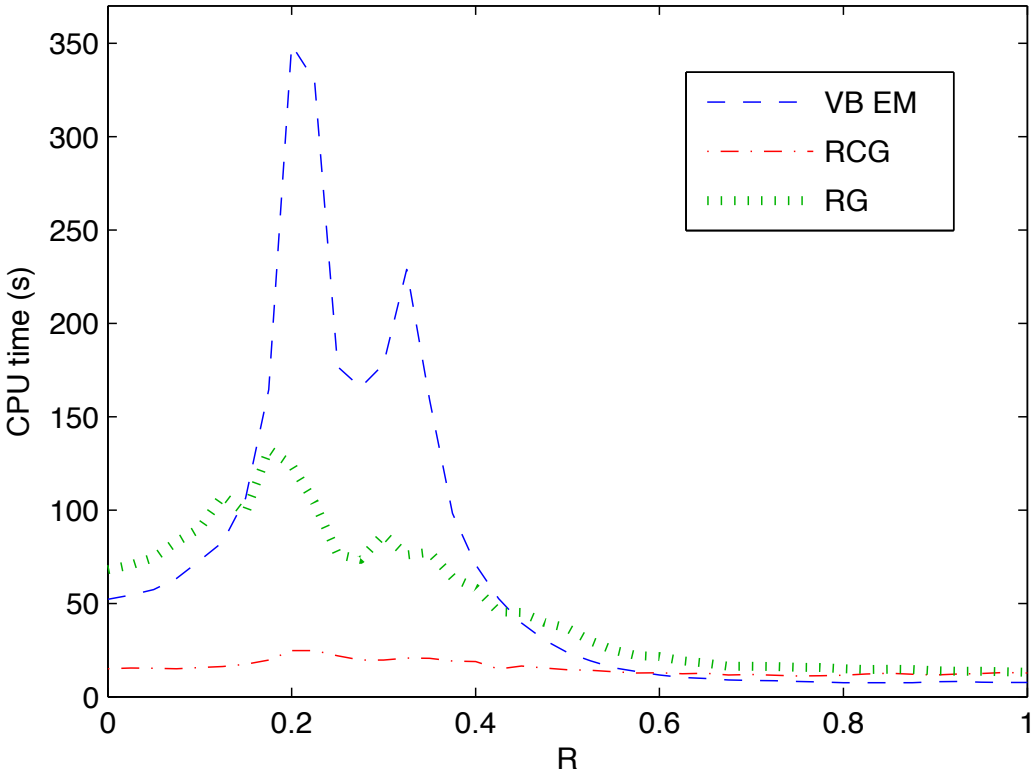

Figure 6: Comparison of VB EM with the Riemannian gradient (RG) and the Riemannian conjugate gradient (RCG) methods using the MoG model with the artificial data. The curves show the median CPU time required for convergence as a function of R. VB EM slows down significantly at the critical overlap of $R \approx 0.2$, while RCG is almost unaffected. A similar slowdown also affects RG implying that the use of conjugate directions contributes to the nearly uniform running time of RCG.

creases. To see this, we set $\varepsilon = 10^{-8}N$ and probed a wide range of values of N. The results are illustrated in Figure 8 which shows that VB EM slows down faster than linearly as the number of samples increases. The most likely reason for this behaviour is that the posterior will be more peaked when the number of observations is large and this slows down the alternating VB EM iteration. The same phenomenon also affects RCG, but the effect is much stronger in VB EM. The results suggest that especially for large data sets, it can be worthwhile to consider alternatives to basic batch VB EM, such as on-line algorithms (Sato, 2001) or gradient-based methods.

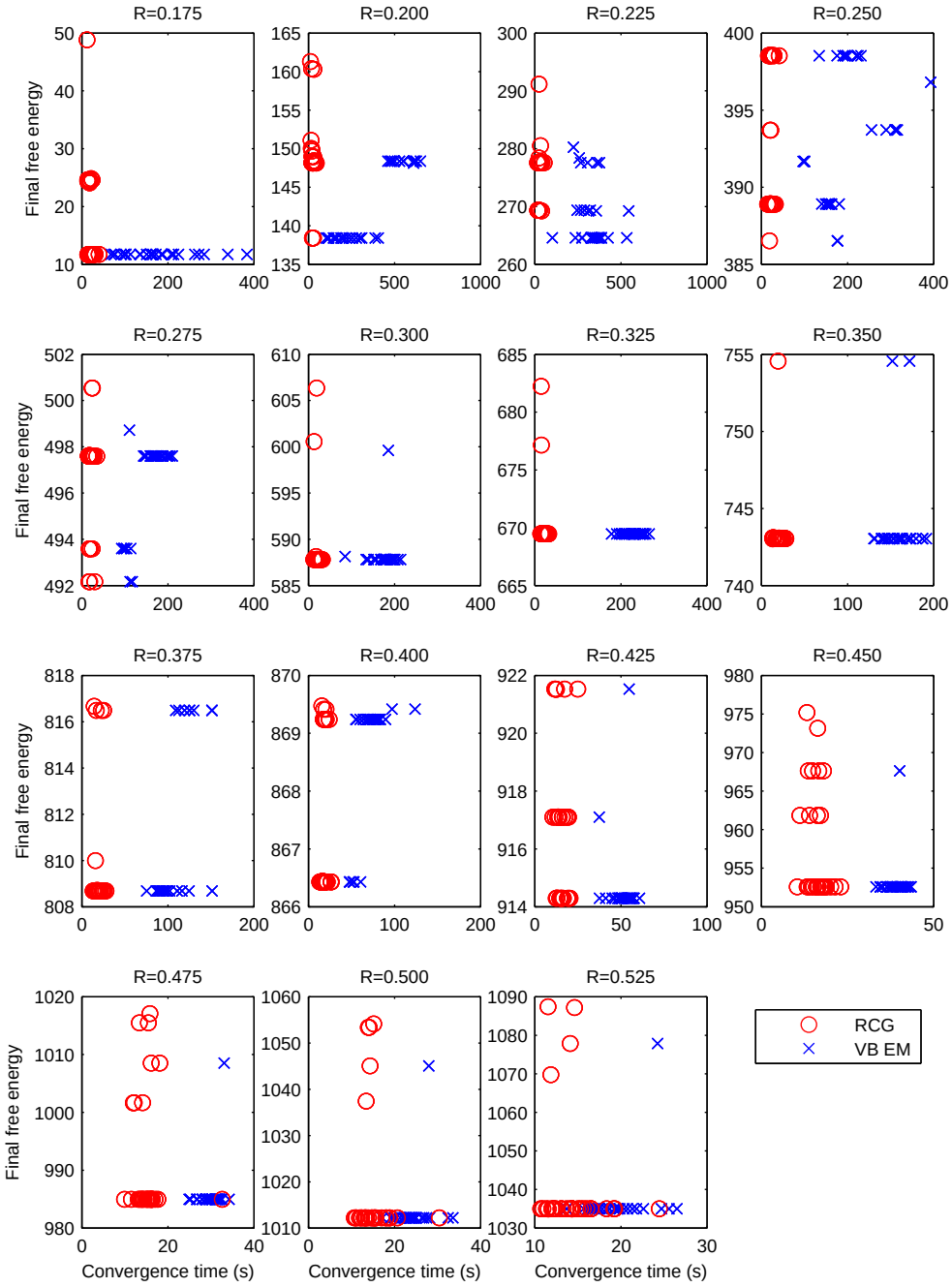

Figure 7: Final free energy value as a function of running time in the critical parameter range in the MoG experiment with varying R. Both RCG and VB EM sometimes fail to converge to the global optimum. Interestingly, there usually is no correlation between the quality of the solution and convergence time.

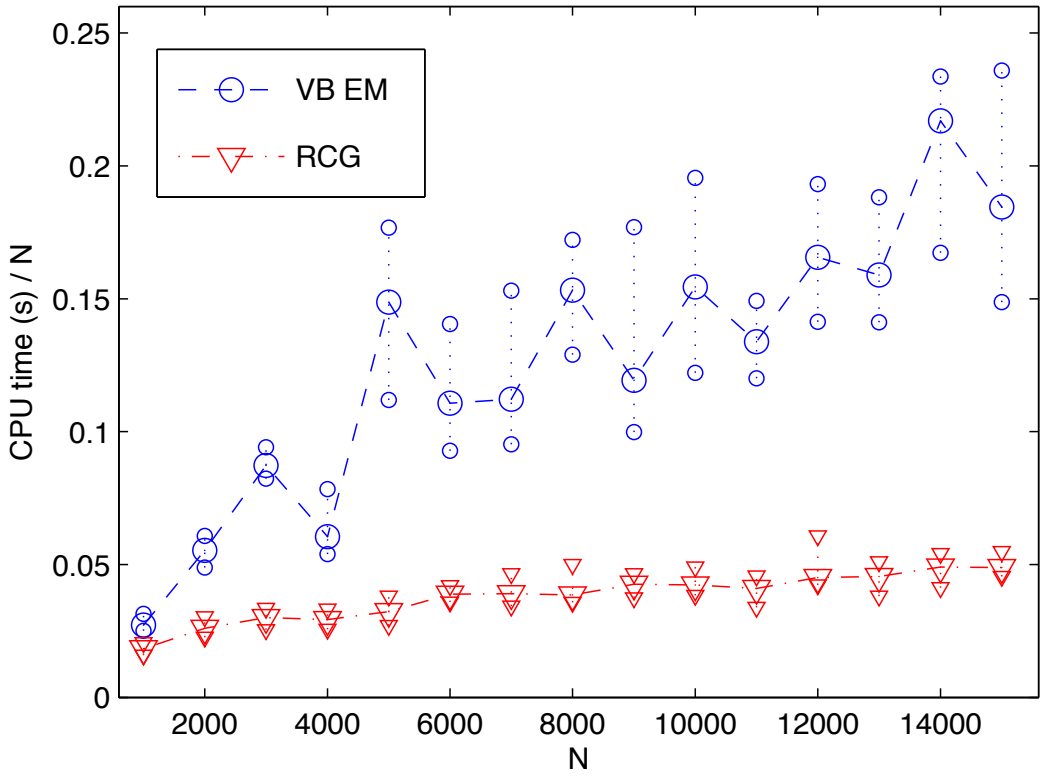

Figure 8: Median convergence times of 30 simulations of the MoG model on artificial data as a function of the number of observations N with VB EM and the Riemannian conjugate gradient (RCG) algorithms. The smaller marks indicate 25% and 75% quantiles.

The model	$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) + \mathbf{n}(t)$ $\mathbf{s}(t) = \mathbf{s}(t-1) + \mathbf{g}(\mathbf{s}(t-1), \boldsymbol{\theta}_\mathbf{g}) + \mathbf{m}(t)$
Key variables	$\boldsymbol{\theta}_S = (\mathbf{s}(t))$, $\boldsymbol{\theta}_\boldsymbol{\theta} = (\boldsymbol{\theta}_\mathbf{f}, \boldsymbol{\theta}_\mathbf{g}, \mathbf{v}_m, \mathbf{v}_n)$
The approximation	$q(\boldsymbol{\theta}_S, \boldsymbol{\theta}_\boldsymbol{\theta}) = q(\boldsymbol{\theta}_S) q(\boldsymbol{\theta}_\boldsymbol{\theta})$, where $q(\boldsymbol{\theta}_\boldsymbol{\theta})$ is Gaussian with diagonal covariance and $q(\boldsymbol{\theta}_S)$ Gaussian with tridiagonal precision (see text)
The update algorithm	Joint RCG updates for the means of $\boldsymbol{\theta}_S, \boldsymbol{\theta}_\mathbf{f}, \boldsymbol{\theta}_\mathbf{g}$, fixed point updates for their covariances, VB EM updates for the rest

Table 2: Summary of the nonlinear state-space model

5. Case Study: Nonlinear State-Space Models

As the second case study, we consider the nonlinear state-space model (NSSM) introduced by Valpola and Karhunen (2002). The model is specified by the generative model

$$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) + \mathbf{n}(t), \tag{17}$$

$$\mathbf{s}(t) = \mathbf{s}(t-1) + \mathbf{g}(\mathbf{s}(t-1), \boldsymbol{\theta}_\mathbf{g}) + \mathbf{m}(t), \tag{18}$$

where t is time, $\mathbf{x}(t)$ are the observations, and $\mathbf{s}(t)$ are the hidden states. The observation mapping $\mathbf{f}$ and the dynamical mapping $\mathbf{g}$ are nonlinear and they are modelled with multi-layer perceptron (MLP) networks whose weight matrices are included in $\boldsymbol{\theta}_\mathbf{f}$ and $\boldsymbol{\theta}_\mathbf{g}$. The observation noise vector $\mathbf{n}$ and process noise vector $\mathbf{m}$ are assumed Gaussian with zero mean and covariances $\mathrm{diag}(\exp(2\mathbf{v}_n))$ and $\mathrm{diag}(\exp(2\mathbf{v}_m))$. The latent states $\mathbf{s}(t)$ are commonly denoted by $\boldsymbol{\theta}_\boldsymbol{S}$. The model parameters include both the weights of the MLP networks and a number of hyperparameters. The posterior approximation of these parameters is a Gaussian with a diagonal covariance matrix. The posterior approximation of the states $q(\boldsymbol{\theta}_\boldsymbol{S}|\boldsymbol{\xi}_\boldsymbol{S})$ is a Gaussian Markov random field with a correlation between the corresponding components of subsequent state vectors $s_j(t)$ and $s_j(t-1)$, as in Equation (13). This is a realistic minimum assumption for modelling the dependence of the state vectors $\mathbf{s}(t)$ and $\mathbf{s}(t-1)$ (Valpola and Karhunen, 2002). Omitted details of the model are presented in Appendix C and a summary is given in Table 2.

Because of the nonlinearities the model is not in the conjugate exponential family, and the standard VB learning methods are only applicable to hyperparameters but not to the latent states or weights of the MLPs. The free energy (1) can nevertheless be evaluated by linearising the MLP networks $\mathbf{f}$ and $\mathbf{g}$ (Honkela and Valpola, 2005; Honkela et al., 2007). This allows evaluating the gradient with respect to $\boldsymbol{\xi}_\boldsymbol{S}$, $\boldsymbol{\xi}_\mathbf{f}$, and $\boldsymbol{\xi}_\mathbf{g}$ and using a gradient based optimiser to adapt the parameters. Combining Equations (4) and (12), the Riemannian gradient for the mean elements is given by

$$\tilde{\nabla}_{\boldsymbol{\mu}_q} \mathcal{F}(\boldsymbol{\xi}) = \boldsymbol{\Sigma}_q \nabla_{\boldsymbol{\mu}_q} \mathcal{F}(\boldsymbol{\xi}),$$

where $\boldsymbol{\mu}_q$ is the mean of the variational approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ and $\boldsymbol{\Sigma}_q$ is the corresponding covariance. The covariance matrix of the model parameters is diagonal while the inverse covariance matrix of the latent states $\mathbf{s}(t)$ is block-diagonal with tridiagonal blocks. This implies that all computations with these can be done in linear time with respect to the number of the parameters. The covariances are updated separately using a fixed-point update rule similar to (8) as described by Valpola and Karhunen (2002). A complete derivation of the free energy of the model is presented in Appendix C.

5.1 Experiments

We applied the method for learning nonlinear state-space models presented above to real world speech data. Experiments were conducted with different data sizes to study the performance differences between the algorithms. The data consisted of 21 dimensional mel-frequency log power speech spectra of continuous human speech. This is a detailed representation of speech signals similar to those often used in speech recognition. A segment of 100 samples corresponds to approximately 0.8 seconds of speech. The task is to learn a nonlinear dynamical model for this data.

To study the performance differences between the Riemannian conjugate gradient (RCG) method, standard Riemannian gradient (RG) method, standard conjugate gradient (CG) method and the heuristic algorithm from Valpola and Karhunen (2002), the algorithms were applied to different sized parts of the speech data set. Unfortunately a reasonable comparison with a VB EM algorithm was impossible because an extended-Kalman-filter-based VB EM algorithm failed with the nonlinear model.

The size of the data subsets varied between 200 and 500 samples. A five-dimensional state-space was used. The MLP networks for the observation and dynamical mappings had 20 hidden nodes. Four different initialisations and two different segments of data of each size were used, resulting in eight repetitions for each algorithm and data size. The results for different data segments of the

same size were pooled together as the convergence times were in general very similar. An algorithm was assumed to have converged when $|\mathcal{F}^t - \mathcal{F}^{t-1}| < \varepsilon = (10^{-5}N/80)$ for 5 consecutive iterations, where $\mathcal{F}^t$ is the free energy at iteration t and N is the size of the data set. Alternatively, the iteration was stopped after 24 hours even if it had not converged.

The MLP network is notoriously prone to local optima. Practically all our simulations converged to local optima with different parameter estimates, but there were no statistically significant differences in the free energies corresponding to these optima attained by different algorithms (Wilcoxon rank-sum test, 5 % significance level). In practice, the free energy values tend to have a very strong correlation with predictive performance of the model (Honkela et al., 2007). There were still some differences, and especially the RG algorithm with smaller data sizes often appeared to converge very early to an extremely poor solution. These were filtered by removing results where the attained free energy that was more than two RCG standard deviations worse than RCG average for the particular data set. Thus all the used results are from runs converging to a roughly equally good solution. The results of one run where the heuristic algorithm diverged were also discarded from the analysis.

The results can be seen in Figure 9. The plain CG and RG methods were clearly slower than others and the maximum runtime was reached by most CG and some RG runs. RCG was clearly the fastest algorithm with the heuristic method of Valpola and Karhunen (2002) between these extremes. The observed differences are, save for a few exceptions mostly with smaller data sets, statistically significant (Wilcoxon rank-sum test, 5 % significance level).

As a more realistic example, a larger data set of 1000 samples was used to train a seven-dimensional state-space model. In this experiment both MLP networks of the NSSM had 30 hidden nodes. The convergence criterion was $\varepsilon = 10^{-6}$ and the maximum runtime was 72 hours. The performances of the RCG, RG, CG methods and the heuristic algorithm were compared. The results can be seen in Figure 10. The results show the convergence for five different initialisations with markers at the end showing when the convergence was reached. It should be noted that the scale of the CPU time axis is logarithmic.

RCG clearly outperformed the other algorithms in this experiment as well. In particular, both RG and CG hit the maximum runtime in every run, and especially CG was nowhere near convergence at this time. RCG also outperformed the heuristic algorithm (Valpola and Karhunen, 2002) by a factor of more than 10.

6. Case Study: Nonlinear Factor Analysis

The model	$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) + \mathbf{n}(t)$ $\mathbf{s}(t) = \mathcal{N}(0, \mathrm{diag}(\exp(2\mathbf{v}_s)))$
Key variables	$\boldsymbol{S} = (\mathbf{s}(t)), \boldsymbol{\theta} = (\boldsymbol{\theta}_\mathbf{f}, \mathbf{v}_s, \mathbf{v}_n)$
The approximation	$q(\boldsymbol{S}, \boldsymbol{\theta}) = q(\boldsymbol{S})q(\boldsymbol{\theta})$, where both $q(\boldsymbol{\theta})$ and $q(\boldsymbol{S})$ are Gaussian with diagonal covariances
The update algorithm	Joint RCG updates for means of $\boldsymbol{S}, \boldsymbol{\theta}_\mathbf{f}$, fixed point update for their covariances, VB EM updates for the rest

Table 3: Summary of the nonlinear factor analysis model

As the final case study, the RCG and RG methods were implemented as extensions to the VB nonlinear factor analysis (NFA) method (Lappalainen and Honkela, 2000; Honkela and Valpola,

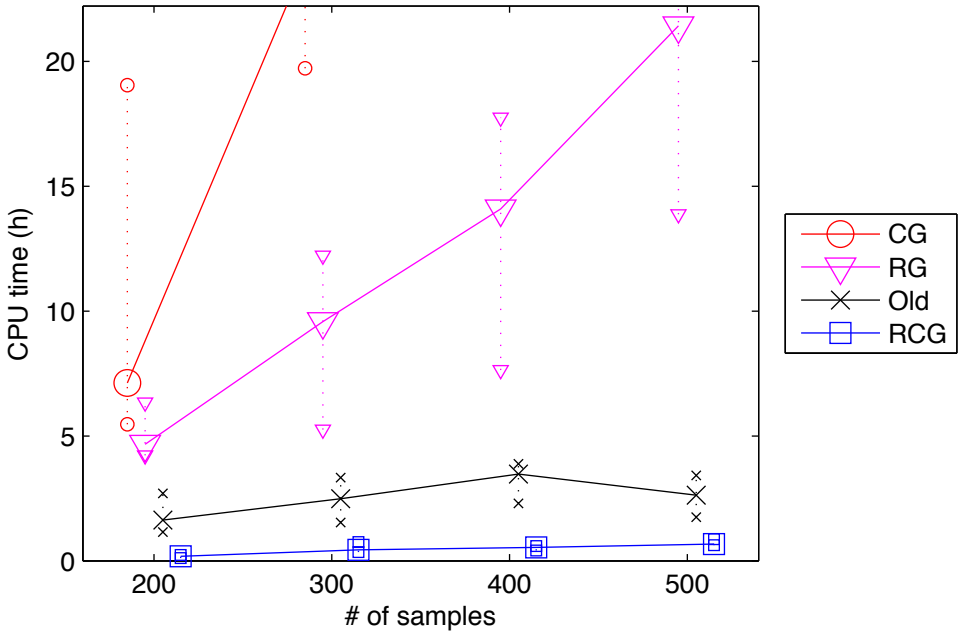

Figure 9: Convergence speed of the Riemannian conjugate gradient (RCG), the Riemannian gradient (RG) and the conjugate gradient (CG) methods as well as the heuristic algorithm (Old) with different data sizes of the speech data set and the nonlinear state-space model. The lines show median times with 25 % and 75 % quantiles shown by the smaller marks. The times were limited to at most 24 hours, which was reached by a number of simulations.

2005; Honkela et al., 2007). NFA models the mapping between latent factors $\mathbf{s}(t)$ and observations $\mathbf{x}(t)$ with an MLP as in Equation (17):

$$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) + \mathbf{n}(t).$$

Instead of the dynamical model of Equation (18), $\mathbf{s}(t)$ has independent Gaussian priors with a unit covariance. As NFA can be seen as a special case of NSSM with no dynamic mapping, the implementation is straightforward. The model is summarised in Table 3. Complete derivation of the model and a learning algorithm based on conjugate gradients is presented by Honkela et al. (2007). The generalisation for Riemannian gradient is straightforward as the Fisher information matrix is diagonal.

The RCG, RG and CG methods were applied for learning an NFA model for parts of the speech data set, a different part of which was also used in the NSSM experiments. As the NFA model cannot capture the dynamics of speech, the experiment aimed at finding a nonlinear embedding of speech on a lower dimensional manifold. To stimulate this, we drew a suitable subset of samples randomly from the full data set of 7860 samples, excluding any dynamical relations in the data. Silent segments were excluded from the data.

We tested each algorithm for data sets ranging in size from 300 to 1000 samples, running 10 simulations with different random initialisations for every setting. The results are shown in Figure

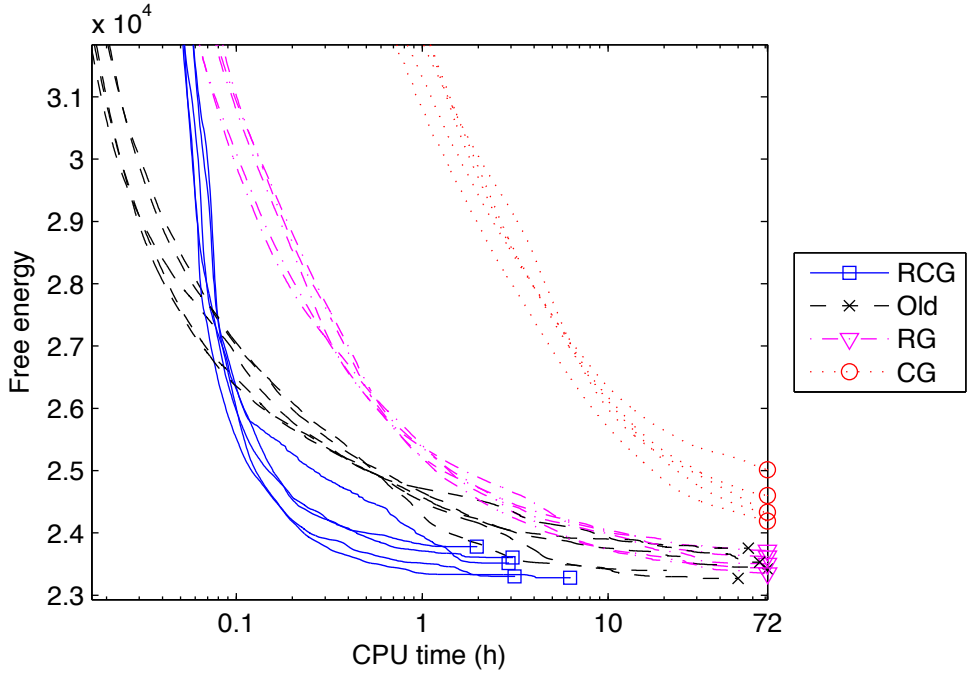

Figure 10: Comparison of the performance of the Riemannian conjugate gradient (RCG), the Riemannian gradient (RG), the conjugate gradient (CG) methods and the heuristic algorithm with the full speech data set of 1000 samples using the nonlinear state-space model. The free energy $\mathcal{F}$ is plotted against computation time using a logarithmic time scale. The tick marks show when simulations either converged or were terminated after 72 hours.

11. RCG is again clearly superior to both RG and CG, but CG is now faster than RG. The observed differences are statistically significant (Wilcoxon rank-sum test, 5 % significance level), except between CG and RG for 300 samples.

7. Discussion

The proposed RCG algorithm combines two improvements over plain gradient optimisation: use of Riemannian gradient and conjugate gradients. One interesting feature in the experimental results is the relative performance of the conjugate gradient and Riemannian gradient algorithms that implement only one of these. Conjugate gradient is faster than Riemannian gradient for NFA, but the opposite is true for NSSM and MoG. Especially the latter differences are quite significant and consistent across several different data sets. One obvious difference between the models is that for NFA the Fisher information matrix is diagonal while for NSSM and MoG this is not the case. This suggests that the Riemannian gradient approach may be the most useful when the metric is more complex, although more careful analysis would be needed to properly understand the effects of different improvements.

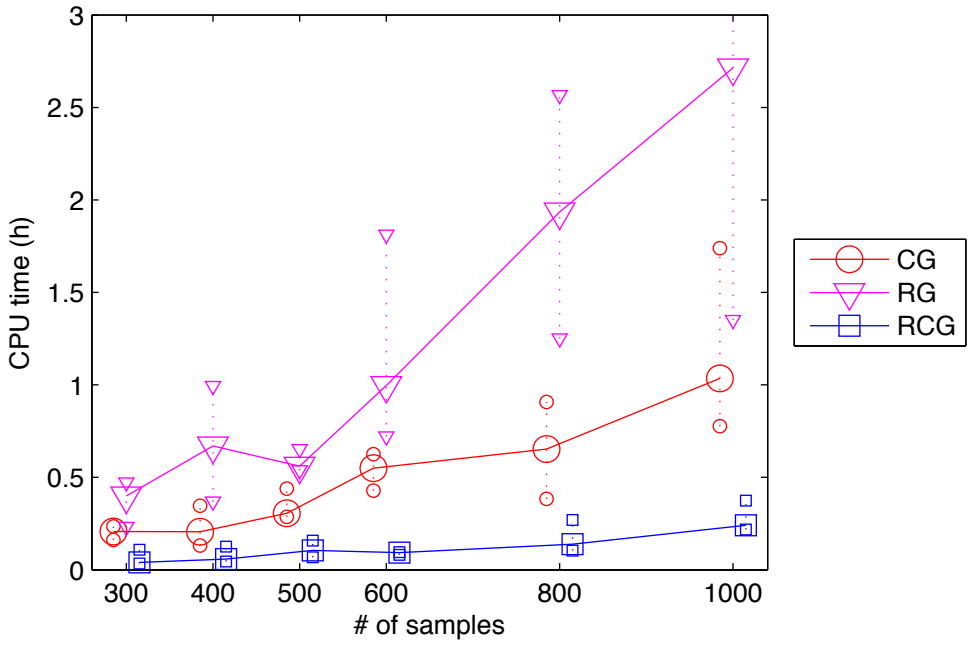

Figure 11: Convergence speed of the Riemannian conjugate gradient (RCG), the Riemannian gradient (RG) and the conjugate gradient (CG) methods with different data sizes of the speech dimensionality reduction data set with the nonlinear factor analysis model. The lines show median times with 25 % and 75 % quantiles shown by the smaller marks.

As illustrated by the MoG example, the RCG algorithm can also be applied to conjugate-exponential models to replace the more common VB EM algorithm. In practice, simpler and more straightforward EM acceleration methods based on, for example, pattern search or adaptive over-relaxation (see, e.g., Honkela et al., 2003; Salakhutdinov and Roweis, 2003) may still provide comparable or better results with less human effort. These methods are only applicable when EM itself is applicable, though.

The experiments in this paper show that using even a greatly simplified variant of the Riemannian conjugate gradient method for some variables is enough to acquire a large speedup. Considering univariate Gaussian distributions, the regular gradient is prone to overemphasise changes to model variables with small posterior variance and underemphasise variables with large posterior variance, as seen from Equations (9)–(11). The posterior variance of latent variables is often much larger than the posterior variance of model parameters and the Riemannian gradient takes this into account in a very natural manner.

The Riemannian conjugate gradient method differs from Euclidean superlinear optimisation methods such as quasi-Newton methods in that it uses higher-order information of the geometry of the parameter space, but not of the function being optimised. These are essentially two independent avenues for improvement: it would be possible, although complicated, to derive a Riemannian quasi-Newton method. Our experiments clearly show that in these problems, a proper model of the

geometry appears significantly more important than using higher-order information of the objective function.

In this paper, we have presented a Riemannian conjugate gradient learning algorithm for fixed-form variational Bayes. The RCG algorithm provides an efficient method for VB learning in models that do not belong to the conjugate-exponential family as required by the standard variational EM algorithm. For suitably structured approximations, the computational overhead from using Riemannian gradients instead of conventional gradients is negligible. In practical examples, the Riemannian gradient approach provided several orders of magnitude speedups over conventional gradient algorithms, thus making VB learning of these models practical on a much larger scale.

MATLAB code for all the models used in the case studies is available at `http://www.cis.hut.fi/projects/bayes/software/ncg/`.

Acknowledgments

This work was supported in part by the Academy of Finland under its Centres of Excellence in Research Program, and the IST Program of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. AH and TR were supported by Postdoctoral Researchers' projects of the Academy of Finland (No 121179, 133145). TR was also supported by the Academy of Finland project "Unsupervised machine learning in latent variable models" (No 121802). This publication only reflects the authors' views.

Appendix A. Convergence of the Riemannian Conjugate Gradient Algorithm

The Riemannian conjugate gradient algorithm has similar superlinear convergence properties to the Euclidean space conjugate gradient algorithm. Assuming the objective $\mathcal{F}(\boldsymbol{\xi})$ has continuous third order derivatives and that there exist $m > 0, M$ such that the Hessian $H(\boldsymbol{\xi})$ satisfies

$$m\mathbf{x}^T\mathbf{x} \leq \mathbf{x}^T H(\boldsymbol{\xi})\mathbf{x} \leq M\mathbf{x}^T\mathbf{x}$$

for all $\boldsymbol{\xi}$ and $\mathbf{x}$, the error decreases quadratically over N steps in an N-dimensional problem. Thus, denoting the optimum by $\boldsymbol{\alpha}$ and the iterates by $\boldsymbol{\xi}_i$, we have (Edelman et al., 1998; Cohen, 1972)

$$||\boldsymbol{\alpha} - \boldsymbol{\xi}_{i+N}|| \leq C||\boldsymbol{\alpha} - \boldsymbol{\xi}_i||^2 \tag{19}$$

in some neighbourhood of $\boldsymbol{\alpha}$.

We now show that the approximations in the RCG algorithm, namely ignoring the parallel transports and performing line searches along straight lines instead of geodesics, do not effect the convergence rate of Equation (19).

Theorem 1 *Assuming the objective $\mathcal{F}(\boldsymbol{\xi})$ has bounded derivatives for up to third order and that the Fisher information $\mathbf{G}(\boldsymbol{\xi})$ is smooth in a neighbourhood of the solution, the RCG algorithm performing a line search along a straight line in the direction*

$$\mathbf{p}_k = -\tilde{\mathbf{g}}_k + \beta\mathbf{p}_{k-1},$$

where $\tilde{\mathbf{g}}_k$ is the Riemannian gradient and

$$\beta = \frac{\langle \tilde{\mathbf{g}}_k, \tilde{\mathbf{g}}_k - \tilde{\mathbf{g}}_{k-1}\rangle_k}{||\tilde{\mathbf{g}}_{k-1}||^2_{k-1}}$$

to find the sequence of iterates $\boldsymbol{\xi}_k$ shares the same convergence property of Equation (19) in some ε-neighbourhood of the solution as the Riemannian conjugate gradient algorithm performing a line search along a geodesic in the direction

$$\mathbf{p}_k^* = -\tilde{\mathbf{g}}_k^* + \beta^* \tau \mathbf{p}_{k-1}^*,$$

where $\tilde{\mathbf{g}}_k$ is the Riemannian gradient, $\tau\mathbf{p}_{k-1}^*$ is the vector $\mathbf{p}_{k-1}^*$ after parallel transport to the starting point of the new search and

$$\beta^* = \frac{\langle \tilde{\mathbf{g}}_k^*, \tilde{\mathbf{g}}_k^* - \tau\tilde{\mathbf{g}}_{k-1}^* \rangle_{k^*}}{||\tilde{\mathbf{g}}_{k-1}^*||^2_{(k-1)^*}},$$

where $\langle \cdot, \cdot \rangle_{k^*}$ is the inner product at $\boldsymbol{\xi}_k^*$, to find the sequence of iterates $\boldsymbol{\xi}_k^*$.

Proof Let us assume that the two algorithms are started at $\boldsymbol{\xi}_k = \boldsymbol{\xi}_k^*$ and $||\boldsymbol{\xi}_k - \boldsymbol{\alpha}|| < \varepsilon$ for some $\varepsilon > 0$. We show by induction on i that $\mathbf{p}_{i-1} = \mathbf{p}_{i-1}^* + O(\varepsilon^2)$ and $\boldsymbol{\xi}_i = \boldsymbol{\xi}_i^* + O(\varepsilon^2)$ for $i \geq k$ which is sufficient to prove the theorem.

The base case is trivial as $\boldsymbol{\xi}_k = \boldsymbol{\xi}_k^*$ and $\mathbf{p}_{k-1} = \mathbf{p}_{k-1}^* = 0$ at the start of the algorithm.

Assume now that the claim is valid for $i = k, \ldots, K$, and let us prove it for $i = K+1$. From Edelman et al. (1998) we know that

$$\begin{aligned} \boldsymbol{\xi}(\varepsilon) &= \boldsymbol{\xi}(0) + \varepsilon\Delta/||\Delta|| + O(\varepsilon^3), \\ \tau\tilde{\mathbf{g}}(\varepsilon) &= \tilde{\mathbf{g}} + O(\varepsilon^2), \end{aligned}$$

where $\boldsymbol{\xi}(\varepsilon)$ is a geodesic in direction Δ, $\tau\tilde{\mathbf{g}}(\varepsilon)$ is the parallel transport of $\tilde{\mathbf{g}}$ to $\boldsymbol{\xi}(\varepsilon)$. The norms of $\tilde{\mathbf{g}}$ and $\mathbf{p}$ are also of the order $O(\varepsilon)$.

The assumption $\boldsymbol{\xi}_K = \boldsymbol{\xi}_K^* + O(\varepsilon^2)$ implies that the gradients evaluated at these points satisfy $\tilde{\mathbf{g}}_K = \tilde{\mathbf{g}}_K^* + O(\varepsilon^2)$. Furthermore,

$$\langle \mathbf{x}, \mathbf{y} \rangle_{K^*} = \mathbf{x}^T \mathbf{G}(\boldsymbol{\xi}_K^*)\mathbf{y} = \mathbf{x}^T(\mathbf{G}(\boldsymbol{\xi}_K) + O(\varepsilon^2))\mathbf{y} = \langle \mathbf{x}, \mathbf{y} \rangle_K + O(\varepsilon^2 ||\mathbf{x}||_K ||\mathbf{y}||_K).$$

Using the above asymptotics,

$$\begin{aligned} \beta^* &= \frac{\langle \tilde{\mathbf{g}}_K^*, \tilde{\mathbf{g}}_K^* - \tau\tilde{\mathbf{g}}_{K-1}^* \rangle_{K^*}}{||\tilde{\mathbf{g}}_{K-1}^*||^2_{(K-1)^*}} = \frac{\langle \tilde{\mathbf{g}}_K + O(\varepsilon^2), \tilde{\mathbf{g}}_K - \tilde{\mathbf{g}}_{K-1} + O(\varepsilon^2) \rangle_{K^*}}{||\tilde{\mathbf{g}}_{K-1} + O(\varepsilon^2)||^2_{(K-1)^*}} \\ &= (1 + O(\varepsilon)) \frac{\langle \tilde{\mathbf{g}}_K, \tilde{\mathbf{g}}_K - \tilde{\mathbf{g}}_{K-1} \rangle_{K^*} + O(\varepsilon^3)}{||\tilde{\mathbf{g}}_{K-1}||^2_{(K-1)^*}} = (1 + O(\varepsilon)) \frac{\langle \tilde{\mathbf{g}}_K, \tilde{\mathbf{g}}_K - \tilde{\mathbf{g}}_{K-1} \rangle_K + O(\varepsilon^3)}{||\tilde{\mathbf{g}}_{K-1}||^2_{(K-1)} + O(\varepsilon^3)} \\ &= (1 + O(\varepsilon)) \left(\frac{\langle \tilde{\mathbf{g}}_K, \tilde{\mathbf{g}}_K - \tilde{\mathbf{g}}_{K-1} \rangle_K}{||\tilde{\mathbf{g}}_{K-1}||^2_{(K-1)}} + O(\varepsilon) \right) = \beta + O(\varepsilon). \end{aligned}$$

Similarly we can find the difference in the search direction,

$$\begin{aligned} \mathbf{p}_K^* &= -\tilde{\mathbf{g}}_K^* + \beta^* \tau \mathbf{p}_{K-1}^* = -\tilde{\mathbf{g}}_K + \beta^* \mathbf{p}_{K-1}^* + O(\varepsilon^2) = -\tilde{\mathbf{g}}_K + \beta \mathbf{p}_{K-1} + O(\varepsilon^2) \\ &= \mathbf{p}_K + O(\varepsilon^2), \end{aligned}$$

which completes the first part of the induction step.

The corresponding step lengths, t^* and t, also differ by $O(\varepsilon)$. To show this, let us use the Taylor expansion of f about the optimum $\boldsymbol{\alpha}$:

$$\begin{aligned}\mathcal{F}(\boldsymbol{\xi}) &= \mathcal{F}(\boldsymbol{\alpha}) + \frac{1}{2}(\boldsymbol{\xi}-\boldsymbol{\alpha})^T H(\boldsymbol{\alpha})(\boldsymbol{\xi}-\boldsymbol{\alpha}) + O(\varepsilon^3),\\ \nabla\mathcal{F}(\boldsymbol{\xi}) &= H(\boldsymbol{\alpha})(\boldsymbol{\xi}-\boldsymbol{\alpha}) + O(\varepsilon^2).\end{aligned}$$

The line search finds the zero of $\mathbf{p}_K \cdot \nabla\mathcal{F}(\boldsymbol{\xi})$ along the line $\boldsymbol{\xi} = \boldsymbol{\xi}_{K-1} + t\mathbf{p}_K$, which yields

$$\mathbf{p}_K^T H(\boldsymbol{\alpha}) \left[(\boldsymbol{\xi}_{K-1} + t\mathbf{p}_K - \boldsymbol{\alpha}) + O(\varepsilon^2)\right] = 0,$$

which can be solved to obtain

$$t = -\frac{\mathbf{p}_K^T H(\boldsymbol{\alpha})(\boldsymbol{\xi}_{K-1}-\boldsymbol{\alpha})}{\mathbf{p}_K^T H(\boldsymbol{\alpha})\mathbf{p}_K} + O(\varepsilon),$$

where we have used the fact that $\mathbf{p}_K^T H(\boldsymbol{\alpha})\mathbf{p}_K \geq m||\mathbf{p}_K||^2$.

Correspondingly, the exact algorithm finds the zero along the geodesic from $\boldsymbol{\xi}^*_{K-1}$ in the direction $\mathbf{p}^*_K$ of $(\tau\mathbf{p}^*_K)^T\nabla\mathcal{F}(\boldsymbol{\xi})$

$$\begin{aligned}&(\tau\mathbf{p}^*_K(t^*||\mathbf{p}^*_K||))^T H(\boldsymbol{\alpha})[\boldsymbol{\xi}^*_{K-1}(t^*||\mathbf{p}^*_K||) - \boldsymbol{\alpha} + O(\varepsilon^2)] =\\ &\qquad (\mathbf{p}^*_K + O(\varepsilon^2))^T H(\boldsymbol{\alpha})[\boldsymbol{\xi}^*_{K-1} + t^*\mathbf{p}^*_K - \boldsymbol{\alpha} + O(\varepsilon^2)] + O(\varepsilon^3) = 0,\end{aligned}$$

where $\boldsymbol{\xi}_{K-1}(t^*||\mathbf{p}^*_K||)$ is the geodesic starting from $\boldsymbol{\xi}^*_{K-1}$ in the direction $\mathbf{p}^*_K$, and $\tau\mathbf{p}^*_K(t^*||\mathbf{p}^*_K||)$ is the parallel transport of $\mathbf{p}^*_K$ along this geodesic. The solution of this equation yields

$$\begin{aligned}t^* &= -\frac{(\mathbf{p}^*_K + O(\varepsilon^2))^T H(\boldsymbol{\alpha})[\boldsymbol{\xi}^*_{K-1} - \boldsymbol{\alpha} + O(\varepsilon^3)]}{(\mathbf{p}^*_K + O(\varepsilon^2))^T H(\boldsymbol{\alpha})\mathbf{p}^*_K} + O(\varepsilon)\\ &= -\frac{(\mathbf{p}_K + O(\varepsilon^2))^T H(\boldsymbol{\alpha})[\boldsymbol{\xi}_{K-1} - \boldsymbol{\alpha} + O(\varepsilon^2)]}{(\mathbf{p}_K + O(\varepsilon^2))^T H(\boldsymbol{\alpha})[\mathbf{p}_K + O(\varepsilon^2)]} + O(\varepsilon)\\ &= -(1+O(\varepsilon))\frac{\mathbf{p}_K^T H(\boldsymbol{\alpha})[\boldsymbol{\xi}_{K-1} - \boldsymbol{\alpha}] + O(\varepsilon^3)}{\mathbf{p}_K^T H(\boldsymbol{\alpha})\mathbf{p}_K} + O(\varepsilon) = t + O(\varepsilon).\end{aligned}$$

Now

$$\begin{aligned}\boldsymbol{\xi}^*_K &= \boldsymbol{\xi}^*_{K-1}(t^*||\mathbf{p}^*_K||) = \boldsymbol{\xi}^*_{K-1} + t^*\mathbf{p}^*_K + O(\varepsilon^3)\\ &= \boldsymbol{\xi}_{K-1} + O(\varepsilon^2) + [t + O(\varepsilon)][\mathbf{p}_K + O(\varepsilon^2)] + O(\varepsilon^3) = \boldsymbol{\xi}_K + O(\varepsilon^2),\end{aligned}$$

which completes the proof. ■

Appendix B. Derivations of the Mixture-of-Gaussians Model

In this section we present details of the variational MoG model, including necessary EM updates, the free energy and the metric tensor for the RCG algorithm.

B.1 VB EM for the Mixture-of-Gaussians Model

This section is completely based on the variational treatment of the MoG model of Attias (2000) and Bishop (2006). Because of this, some details of the derivation of the VB EM algorithm for the MoG model will be omitted here and we will concentrate only on the most important results.

In expressing the update rules for the distribution parameters in Equations (14), (15) and (16), we will find the following definitions useful:

$$
\begin{aligned}
N_k &= \sum_{n=1}^{N} r_{nk}, \\
\bar{\mathbf{x}}_k &= \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n, \\
\mathbf{S}_k &= \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \bar{\mathbf{x}}_k)(\mathbf{x}_n - \bar{\mathbf{x}}_k)^T, \\
\log \tilde{\Lambda}_k &= \sum_{i=1}^{D} \psi\left(\frac{\nu_k + 1 - i}{2} \right) + D \log 2 + \log |\mathbf{W}_k|, \\
\log \tilde{\pi}_k &= \psi(\alpha_k) - \psi\left(\sum_{k'=1}^{K} \alpha_{k'} \right),
\end{aligned}
$$

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

$$
\psi(x) = \frac{d}{dx} \log \Gamma(x).
$$

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

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

where

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

The parameters r_{nk} are called *responsibilities* because they represent the responsibility the kth component takes in explaining the nth observation. The responsibilities can be arranged into a matrix $\mathbf{R} = (r_{nk})$ and will have to satisfy the following conditions

$$
0 \leq r_{nk} \leq 1, \tag{20}
$$

$$
\sum_{k=1}^{K} r_{nk} = 1. \tag{21}
$$

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

$$\alpha_k = \alpha_0 + N_k, \tag{22}$$
$$\beta_k = \beta_0 + N_k, \tag{23}$$
$$\nu_k = \nu_0 + N_k + 1, \tag{24}$$
$$\mathbf{m}_k = \frac{1}{\beta_0 + N_k}(\beta_0 \mathbf{m}_0 + N_k \bar{\mathbf{x}}_k),$$
$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + N_k \mathbf{S}_k + \frac{\beta_0 N_k}{\beta_0 + N_k}(\bar{\mathbf{x}}_k - \mathbf{m}_0)(\bar{\mathbf{x}}_k - \mathbf{m}_0)^T. \tag{25}$$

B.2 The Free Energy

The free energy of Equation (1) is

$$\begin{aligned}
\mathcal{F} &= \sum_{\mathbf{Z}} \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\Lambda}} q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) \log \frac{q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})}{p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})} d\boldsymbol{\pi} d\boldsymbol{\mu} d\boldsymbol{\Lambda} \\
&= E_q\{\log q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})\} - E_q\{\log p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})\} \\
&= E_q\{\log q(\mathbf{Z})\} + E_q\{\log q(\boldsymbol{\pi})\} + E_q\{\log q(\boldsymbol{\mu}, \boldsymbol{\Lambda})\} \\
&\quad - E_q\{\log p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})\} - E_q\{\log p(\mathbf{Z}|\boldsymbol{\pi})\} \\
&\quad - E_q\{\log p(\boldsymbol{\pi})\} - E_q\{\log p(\boldsymbol{\mu}, \boldsymbol{\Lambda})\}.
\end{aligned} \tag{26}$$

These expectations can be evaluated to give (Bishop, 2006)

$$E_q\{\log q(\mathbf{Z})\} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \log r_{nk},$$
$$E_q\{\log q(\boldsymbol{\pi})\} = \sum_{k=1}^{K} (\alpha_k - 1) \log \tilde{\pi}_k + \log C(\boldsymbol{\alpha}),$$
$$E_q\{\log q(\boldsymbol{\mu}, \boldsymbol{\Lambda})\} = \sum_{k=1}^{K} \left\{ \frac{1}{2} \log \tilde{\Lambda}_k + \frac{D}{2} \log \frac{\beta_k}{2\pi} - \frac{D}{2} - H_q\{\boldsymbol{\Lambda}_k\} \right\},$$
$$E_q\{\log p(\mathbf{Z}|\boldsymbol{\pi})\} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \log \tilde{\pi}_k,$$
$$E_q\{\log p(\boldsymbol{\pi})\} = \log C(\boldsymbol{\alpha}_0) + (\alpha_0 - 1) \sum_{k=1}^{K} \log \tilde{\pi}_k,$$

$$E_q\{\log p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})\} = \frac{1}{2} \sum_{k=1}^{K} N_k \left\{ \log \tilde{\Lambda}_k - \frac{D}{\beta_k} - \nu_k \operatorname{Tr}(\mathbf{S}_k \mathbf{W}_k) - \nu_k (\bar{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\bar{\mathbf{x}}_k - \mathbf{m}_k) - D \log 2\pi \right\},$$

$$E_q\{\log p(\boldsymbol{\mu}, \boldsymbol{\Lambda})\} = \frac{1}{2} \sum_{k=1}^{K} \left\{ D \log \frac{\beta_0}{2\pi} + \log \tilde{\Lambda}_k - \frac{D\beta_0}{\beta_k} - \beta_0 \nu_k (\mathbf{m}_k - \mathbf{m}_0)^T \mathbf{W}_k (\mathbf{m}_k - \mathbf{m}_0) \right\}$$
$$+ K \log B(\mathbf{W}_0, \nu_0) + \frac{\nu_0 - D - 1}{2} \sum_{k=1}^{K} \log \tilde{\Lambda}_k - \frac{1}{2} \sum_{k=1}^{K} \nu_k \operatorname{Tr}(\mathbf{W}_0^{-1} \mathbf{W}_k),$$

where $\mathrm{Tr}(\mathbf{A})$ denotes the trace of matrix $\mathbf{A}$ and $H_q\{\mathbf{\Lambda}_k\}$ is the entropy of the distribution $q(\mathbf{\Lambda}_k)$. The functions C and B are defined by the following two equations:

$$C(\boldsymbol{\alpha}) = \Gamma\left(\sum_{k=1}^{K}\alpha_k\right)\left(\prod_{k=1}^{K}\Gamma(\alpha_k)\right)^{-1},$$

$$B(\mathbf{W},\nu) = |\mathbf{W}|^{-\nu/2}\left(2^{\nu D/2}\pi^{D(D-1)/4}\prod_{i=1}^{D}\Gamma\left(\frac{\nu+1-i}{2}\right)\right)^{-1}.$$

B.3 Riemannian Conjugate Gradient for the Mixture-of-Gaussians Model

To be able to compare the VB EM and RCG algorithms, we assume that the approximate posterior distribution $q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\mathbf{\Lambda})$ takes the same functional form as in the case of the VB EM algorithm. Thus, the fixed form posterior distributions are given by Equations (14), (15) and (16) and the free energy which is to be minimised by the RCG algorithm is given Equation (26). In this work, we will only be optimising the responsibilities r_{nk} and the means $\mathbf{m}_k$ using gradient-based methods. All 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 (22), (23), (24) and (25).

There are a few things that have to be taken into account when deriving gradient-based algorithms for the MoG model. Firstly, the responsibilities have to satisfy the constraints given by Equations (20) and (21). This can be enforced by using the *softmax* parametrisation

$$r_{nk} = \frac{e^{\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl}}}. \tag{27}$$

It can be easily seen that by using this parametrisation the responsibilities are always positive and $\sum_{k=1}^{K} r_{nk} = 1$. As a results it holds that $0 \leq r_{nk} \leq 1$ and we can conduct unconstrained optimisation in the $\boldsymbol{\gamma}$ space.

Secondly, if we set the responsibilities $r_{nk}, n = 1\ldots N, k = 1\ldots K-1$ to some values, the values of $r_{nK}, n = 1\ldots N$ are given by condition (21), that is $r_{nK} = 1 - \sum_{k=1}^{K-1} r_{nk}$. 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 parametrisation (27), this means that we can regard the parameters γ_{nK} as constants and only optimise the free energy with respect to parameters $\gamma_{nk}, n = 1\ldots N, k = 1\ldots K-1$. This is especially important when using the Riemannian gradient.

The gradient of the free energy (26) with respect to $\mathbf{m}_k$ is given by

$$\nabla_{\mathbf{m}_k}\mathcal{F} = \nu_k\mathbf{W}_k(N_k(\mathbf{m}_k - \bar{\mathbf{x}}_k) + \beta_0(\mathbf{m}_k - \mathbf{m}_0))$$

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

$$\frac{\partial\mathcal{F}}{\partial\gamma_{nk}} = E_{nk} - r_{nk}F_n,$$

where

$$E_{nk} = r_{nk}\left(\log r_{nk} - \log\tilde{\pi}_k - \frac{1}{2}\left(\log\tilde{\Lambda}_k - \frac{D}{\beta_k} - D\log 2\pi - \nu_k(\mathbf{x}_n - \mathbf{m}_k)^T\mathbf{W}_k(\mathbf{x}_n - \mathbf{m}_k)\right)\right)$$

and

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

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

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

where r'_{nk} is the new responsibility, $\Delta\gamma_{nk}$ is the change in parameter γ_{nk} determined by a line search in the search direction and c_n is a normalising 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

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

In order to use the Riemannian gradient, we need to know the Riemannian metric tensor $\mathbf{G}$ of the parameter space $(\mathbf{m}, \boldsymbol{\gamma})$ which is given by Equation (3). The resulting matrix is a block diagonal matrix with blocks $\mathbf{A}_k = \beta_k \nu_k \mathbf{W}_k$ for each $\mathbf{m}_k$ and blocks $\mathbf{B}_n = -\mathbf{r}_n^T \mathbf{r}_n + \mathrm{diag}(\mathbf{r}_n)$ for each sample, where $\mathbf{r}_n$ is the nth row of the responsibility matrix $\mathbf{R}$ except for element r_{nK}, that is $\mathbf{r}_n = [r_{n1} \cdots r_{nK-1}]$. $\mathrm{diag}(\mathbf{a})$ is used here to denote a square matrix which has the elements of vector $\mathbf{a}$ on its main diagonal. This block-diagonal structure of the matrix makes the Riemannian vector operations easy and efficient to implement.

Since the EM updates of parameters are computationally efficient compared to the evaluation of the objective function, it is more efficient to do them also within the line search of the RCG update rather than as a separate step as in Algorithm 1.

Appendix C. Derivation of the Nonlinear State-Space Model

In this section we review the details of the nonlinear state-space model of Valpola and Karhunen (2002).

C.1 Probability Model and Priors

The nonlinear state-space model of Valpola and Karhunen (2002) can be described with these two equations:

$$\mathbf{s}(t) \sim \mathcal{N}(\mathbf{s}(t-1) + \mathbf{g}(\mathbf{s}(t-1), \boldsymbol{\theta}_\mathbf{g}), \mathrm{diag}(\exp(2\mathbf{v}_m))), \tag{28}$$

$$\mathbf{x}(t) \sim \mathcal{N}(\mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}), \mathrm{diag}(\exp(2\mathbf{v}_n))), \tag{29}$$

where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. The nonlinear mappings are modelled with MLP networks:

$$\begin{aligned} \mathbf{g}(\mathbf{s}(t-1), \boldsymbol{\theta}_\mathbf{g}) &= \mathbf{D} \tanh[\mathbf{C}\mathbf{s}(t-1) + \mathbf{c}] + \mathbf{d}, \\ \mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) &= \mathbf{B} \tanh[\mathbf{A}\mathbf{s}(t) + \mathbf{a}] + \mathbf{b}. \end{aligned}$$

The model for data $\mathbf{X}$ is thus described using unobserved variables

$$\boldsymbol{\theta} = (\mathbf{s}(t), \mathbf{A}, \mathbf{a}, \mathbf{B}, \mathbf{b}, \mathbf{C}, \mathbf{c}, \mathbf{D}, \mathbf{d}, \mathbf{v}_m, \mathbf{v}_n).$$

The priors of the variables are specified to fix the scaling ambiguity between $\mathbf{s}$ and $\mathbf{A}$ and to have a hierarchical prior allowing automatic relevance determination (ARD) (Bishop, 2006) like decisions to inactivate parts of the model:

$$\begin{aligned}
A_{ij} &\sim \mathcal{N}(0,1),\\
\Phi_{ij} &\sim \mathcal{N}(0, \exp(2v_{\Phi_j})),\\
\phi_i &\sim \mathcal{N}(m_\phi, \exp(2v_\phi)),\\
v_{\nu_i} &\sim \mathcal{N}(m_{v_\nu}, \exp(2v_{v_\nu})),\\
v_{\Phi_j} &\sim \mathcal{N}(m_{v_\Phi}, \exp(2v_{v_\Phi})),
\end{aligned}$$

where $\nu \in \{m,n\}$, $\phi \in \{a,b,c,d\}$ and $\Phi \in \{B,C,D\}$. All the hyperparameters have vague priors $\mathcal{N}(0,100^2)$.

C.2 Posterior Approximation

In order to allow efficient learning, the posterior approximation $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Lambda}_{\boldsymbol{\theta}}^{-1})$ is restricted to be Gaussian with mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and precision (inverse covariance) $\boldsymbol{\Lambda}_{\boldsymbol{\theta}}$. Furthermore, the precision of the approximation is restricted to be almost diagonal. The only allowed off-diagonal terms are in the approximation of $\mathbf{s}(t)$ which includes a correlation between $s_i(t)$ and $s_i(t+1)$. Different components of the state vector $\mathbf{s}(t)$ are still assumed independent, and the posterior approximation of the states is a product of independent chains.

Following the theory of Gaussian Markov random fields, this assumption translates to a tridiagonal precision (inverse covariance) matrix with non-zero elements only on the main diagonal and on the diagonal corresponding to the assumed links. The corresponding covariance matrix has full blocks for each component of the state.

C.3 The Free Energy

In order to derive the value of the free energy (1), we note that

$$\mathcal{F}(q(\boldsymbol{\theta})) = E_{q(\boldsymbol{\theta})}\{\log q(\boldsymbol{\theta})\} + E_{q(\boldsymbol{\theta})}\{-\log p(\mathbf{X}, \boldsymbol{\theta})\}.$$

The first term is the negative entropy of a Gaussian

$$E_{q(\boldsymbol{\theta})}\{\log q(\boldsymbol{\theta})\} = -\frac{N}{2}\log(2\pi e) - \frac{1}{2}\log\det\boldsymbol{\Lambda}_{\boldsymbol{\theta}},$$

where N is the dimensionality of $\boldsymbol{\theta}$. The second term splits to a sum of a number of terms according to Equations (28)–(29).

$$E_{q(\boldsymbol{\theta})}\{-\log p(\mathbf{X}, \boldsymbol{\theta})\} = \sum_{t,i} E_{q(\boldsymbol{\theta})}\{-\log p(x_i(t)|\boldsymbol{\theta})\} + \sum_{\gamma \in \boldsymbol{\theta}} E_{q(\boldsymbol{\theta})}\left\{-\log p(\gamma|\boldsymbol{\theta}_{\setminus\gamma})\right\},$$

where $\boldsymbol{\theta}_{\setminus\gamma}$ denotes the parameters γ depends on.

The terms in the sum are expectations for parameters γ following a normal model $\mathcal{N}(m, e^{2v})$. The negative logarithm of the pdf is

$$-\log p(\gamma|\boldsymbol{\theta}_{\setminus\gamma}) = \frac{1}{2}\log(2\pi) + v + \frac{1}{2}(\gamma - m)^2 \exp(-2v).$$

Assuming independent Gaussian approximations[2] for γ, m and v with means $\bar{\gamma}, \bar{m}, \bar{v}$ and variances $\tilde{\gamma}, \tilde{m}, \tilde{v}$, respectively, the expectation is

$$E_{q(\boldsymbol{\theta})}\left\{-\log p(\gamma|\boldsymbol{\theta}_{\setminus\gamma})\right\} = \frac{1}{2}\log(2\pi) + \bar{v} + \frac{1}{2}[(\bar{\gamma} - \bar{m})^2 + \tilde{\gamma} + \tilde{m}]\exp(2\tilde{v} - 2\bar{v}).$$

For the observations $x_i(t)$ we obtain similarly

$$E_{q(\boldsymbol{\theta})}\{-\log p(x_i(t)|\boldsymbol{\theta})\} = \frac{1}{2}\log(2\pi) + \bar{v}_{n_i} + \frac{1}{2}[(x - \bar{f}_i(t))^2 + \tilde{f}_i(t)]\exp(2\tilde{v}_{n_i} - 2\bar{v}_{n_i}),$$

where the means $\bar{f}_i(t)$ and variances $\tilde{f}_i(t)$ of $\mathbf{f}(\mathbf{s}(t))$ are evaluated as explained in Honkela and Valpola (2005); Honkela et al. (2007).

For the states $s_i(t)$ we can similarly derive (Valpola and Karhunen, 2002)

$$\begin{aligned} E_{q(\boldsymbol{\theta})}\left\{-\log p(s_i(t)|\boldsymbol{\theta}_{\setminus\mathbf{s}(t)})\right\} &= \frac{1}{2}\log(2\pi) + \bar{v}_{m_i} \\ &+ \frac{1}{2}\left[(\bar{s}_i(t) - \bar{g}_i(t))^2 + \tilde{s}_i(t) + \tilde{g}_i(t) - 2\check{s}_i(t, t-1)\frac{g_i(t)}{s_i(t-1)}\tilde{s}_i(t-1)\right]\exp(2\tilde{v}_{m_i} - 2\bar{v}_{m_i}), \end{aligned}$$

where $\bar{g}_i(t)$ and $\tilde{g}_i(t)$ are the mean and variance of $\mathbf{g}(\mathbf{s}(t-1))$ evaluated similarly as those of $\mathbf{f}(\mathbf{s}(t))$, and $\check{s}_i(t, t-1)$ is the linear correlation between $s_i(t-1)$ and $s_i(t)$ as explained in Valpola and Karhunen (2002). The partial derivative $\frac{g_i(t)}{s_i(t-1)}$ is evaluated naturally as a by-product of evaluation of $\tilde{g}_i(t)$ as explained previously (Honkela and Valpola, 2005; Honkela et al., 2007).

C.4 Update Rules

The hyperparameters $v_{\Phi_j}, m_\phi, v_\phi, v_{\nu_i}, m_{v_\nu}, v_{v_\nu}, v_{\Phi_j}, m_{v_\Phi}, v_{v_\Phi}$ are updated using a VB EM type scheme to find a global optimum, given current values of the other parameters (Lappalainen and Miskin, 2000). The variances of the states and the weights of the MLP networks are updated using the fixed-point rule and the means by the RCG algorithm, as described in Section 5.

References

S. Amari. *Differential-Geometrical Methods in Statistics*, volume 28 of *Lecture Notes in Statistics*. Springer-Verlag, Berlin, 1985.

S. Amari. Information geometry of the EM and em algorithms for neural networks. *Neural Networks*, 8(9):1379–1408, 1995. doi: 10.1016/0893-6080(95)00003-8.

S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998. doi: 10.1162/089976698300017746.

2. We will use the notation $\bar{\gamma}$ for the mean and $\tilde{\gamma}$ for the variance of γ in the approximation for all variables.

S. Amari and H. Nagaoka. *Methods of Information Geometry*, volume 191 of *Translations of Mathematical Monographs*. American Mathematical Society, Providence, RI, USA, 2000.

C. Archambeau, M. Opper, Y. Shen, D. Cornford, and J. Shawe-Taylor. Variational inference for diffusion processes. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 17–24. MIT Press, Cambridge, MA, USA, 2008.

H. Attias. A variational Bayesian framework for graphical models. In S. Solla, T. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 209–215. MIT Press, Cambridge, MA, USA, 2000.

D. Barber and C. Bishop. Ensemble learning for multi-layer networks. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 395–401. The MIT Press, Cambridge, MA, USA, 1998.

C. M. Bishop. *Pattern Recognition and Machince Learning*. Springer, New York, NY, USA, 2006.

R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995. doi: 10.1137/0916069.

P. Carbonetto. A MATLAB interface for L-BFGS-B. `http://people.cs.ubc.ca/~pcarbo/lbfgsb-for-matlab.html`, March 2007.

A. I. Cohen. Rate of convergence of several conjugate gradient algorithms. *SIAM Journal of Numerical Analysis*, 9(2):248–259, 1972. doi: 10.1137/0709024.

A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998. doi: 10.1137/S0895479895290954.

Z. Ghahramani and M. Beal. Propagation algorithms for variational Bayesian learning. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 507–513. The MIT Press, Cambridge, MA, USA, 2001.

M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *J. of the Royal Statistical Society, Series B (Methodological)*, 2011. In press.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, USA, 3rd edition, 1996.

A. González and J. R. Dorronsoro. Natural conjugate gradient training of multilayer perceptrons. *Neurocomputing*, 71(13–15):2499–2506, 2008. doi: 10.1016/j.neucom.2007.11.035.

A. Honkela and H. Valpola. Unsupervised variational Bayesian learning of nonlinear models. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 593–600. MIT Press, Cambridge, MA, USA, 2005.

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. doi: 10.1023/A:1023655202546.

A. Honkela, H. Valpola, A. Ilin, and J. Karhunen. Blind separation of nonlinear mixtures by variational Bayesian learning. *Digital Signal Processing*, 17(5):914–934, 2007. doi: 10.1016/j.dsp.2007.02.009.

A. Honkela, M. Tornio, T. Raiko, and J. Karhunen. Natural conjugate gradient in variational inference. In *Proceedings of the 14th International Conference on Neural Information Processing (ICONIP 2007)*, volume 4985 of *Lecture Notes in Computer Science*, pages 305–314, Kitakyushu, Japan, 2008. Springer-Verlag, Berlin. doi: 10.1007/978-3-540-69162-4_32.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In M. Jordan, editor, *Learning in Graphical Models*, pages 105–161. The MIT Press, Cambridge, MA, USA, 1999.

M. Kuusela, T. Raiko, A. Honkela, and J. Karhunen. A gradient-based algorithm competitive with variational Bayesian EM for mixture of Gaussians. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2009*, pages 1688–1695, Atlanta, GA, USA, June 2009. doi: 10.1109/IJCNN.2009.5178726.

H. Lappalainen and A. Honkela. Bayesian nonlinear independent component analysis by multilayer perceptrons. In M. Girolami, editor, *Advances in Independent Component Analysis*, pages 93–121. Springer-Verlag, Berlin, 2000.

H. Lappalainen and J. Miskin. Ensemble learning. In M. Girolami, editor, *Advances in Independent Component Analysis*, pages 75–92. Springer-Verlag, Berlin, 2000.

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

J. Nocedal. Theory of algorithms for unconstrained optimization. *Acta Numerica*, 1:199–242, 1992. doi: 10.1017/S0962492900002270.

M. Opper and C. Archambeau. The variational Gaussian approximation revisited. *Neural Computation*, 21(3):786–792, 2009. doi: 10.1162/neco.2008.08-07-592.

M. J. D. Powell. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 12(1):241–254, 1977. doi: 10.1007/BF01593790.

T. Raiko, H. Valpola, M. Harva, and J. Karhunen. Building blocks for variational Bayesian learning of latent variable models. *Journal of Machine Learning Research*, 8(Jan):155–201, January 2007.

R. Salakhutdinov and S. T. Roweis. Adaptive overrelaxed bound optimization methods. In *Proc. 20th International Conference on Machine Learning (ICML 2003)*, pages 664–671. AAAI Press, 2003.

M. Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681, 2001. doi: 10.1162/089976601750265045.

M. Seeger. Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers. In S. Solla, T. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 603–609. MIT Press, Cambridge, MA, USA, 2000.

S. T. Smith. *Geometric Optimization Methods for Adaptive Filtering*. PhD thesis, Harvard University, Cambridge, MA, USA, 1993.

T. Tanaka. Information geometry of mean-field approximation. In Manfred Opper and David Saad, editors, *Advanced Mean Field Methods: Theory and Practice*, pages 259–273. The MIT Press, Cambridge, MA, USA, 2001.

M. K. Titsias and N. D. Lawrence. Bayesian Gaussian process latent variable model. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Workshop on Artificial Intelligence and Statistics*, volume 9, pages 844–851, Chia Laguna Resort, Sardinia, Italy, 2010. JMLR W&CP 9.

H. Valpola. *Bayesian Ensemble Learning for Nonlinear Factor Analysis*. PhD thesis, Helsinki University of Technology, Espoo, Finland, 2000. Published in Acta Polytechnica Scandinavica, Mathematics and Computing Series No. 108.

H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 14(11):2647–2692, 2002. doi: 10.1162/089976602760408017.

H. Valpola, M. Harva, and J. Karhunen. Hierarchical models of variance sources. *Signal Processing*, 84(2):267–282, 2004. doi: 10.1016/j.sigpro.2003.10.014.

J. Winn and C. M. Bishop. Variational message passing. *Journal of Machine Learning Research*, 6: 661–694, April 2005.