Natural Conjugate Gradient in Variational Inference

Antti Honkela, Matti Tornio, Tapani Raiko, and Juha Karhunen

Adaptive Informatics Research Centre, Helsinki University of Technology
P.O. Box 5400, FI-02015 TKK, Finland
{Antti.Honkela, Matti.Tornio, Tapani.Raiko, Juha.Karhunen}@tkk.fi
http://www.cis.hut.fi/projects/bayes/

Abstract. Variational methods for approximate inference in machine learning often adapt a parametric probability distribution to optimize a given objective function. This view is especially useful when applying variational Bayes (VB) to models outside the conjugate-exponential family. For them, variational Bayesian expectation maximization (VB EM) algorithms are not easily available, and gradient-based methods are often used as alternatives. Traditional natural gradient methods use the Riemannian structure (or geometry) of the predictive distribution to speed up maximum likelihood estimation. We propose using the geometry of the variational approximating distribution instead to speed up a conjugate gradient method for variational learning and inference. The computational overhead is small due to the simplicity of the approximating distribution. Experiments with real-world speech data show significant speedups over alternative learning algorithms.

1 Introduction

Variational Bayesian (VB) methods provide an efficient and often sufficiently accurate deterministic approximation to exact Bayesian learning [1]. 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.

Nevertheless, there are many interesting more complicated models which are not in the conjugate exponential family. Similar variational approximations have been applied for many such models [2–7]. The approximating distribution $q(\boldsymbol{\theta}|\boldsymbol{\xi})$, where $\boldsymbol{\theta}$ includes both model parameters and latent variables, is often restricted to be Gaussian with a somehow restricted covariance. Values of the variational parameters $\boldsymbol{\xi}$ can be found by using a gradient-based optimization algorithm.

When applying a generic optimization algorithm for such problem, 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 they can change the influence of other parameters. This implies that the geometry of the problem is in most cases not Euclidean.

Information geometry studies the Riemannian geometric structure of the manifold of probability distributions [8]. It has been applied to derive efficient

natural gradient learning rules for maximum likelihood algorithms in independent component analysis (ICA) and multilayer perceptron (MLP) networks [9]. The approach has been used in several other problems as well, for example in analyzing the properties of an on-line variational Bayesian EM method [10].

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 mean field type VB. This is in contrast with the traditional natural gradient learning [9] which uses the Riemannian structure of the predictive distribution $p(\boldsymbol{X}|\boldsymbol{\theta})$. The proposed method can be used to jointly optimize all the parameters $\boldsymbol{\xi}$ of the approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$, or in conjunction with VB EM for some parameters. The method is especially useful for models that are not in the conjugate exponential family, such as nonlinear models [2–5, 7] or non-conjugate variance models [6] that may not have a tractable exact VB EM algorithm.

2 Variational Bayes

Variational Bayesian learning [1, 5] 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 unknown 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 variable). The approximation is fitted by maximizing a lower bound on marginal log-likelihood

$$\mathcal{B}(q(\boldsymbol{\theta}|\boldsymbol{\xi})) = \left\langle \log \frac{p(\boldsymbol{X},\boldsymbol{\theta})}{q(\boldsymbol{\theta}|\boldsymbol{\xi})} \right\rangle = \log p(\boldsymbol{X}) - D_{\mathrm{KL}}(q(\boldsymbol{\theta}|\boldsymbol{\xi}) \| p(\boldsymbol{\theta}|\boldsymbol{X})), \qquad (1)$$

where $\langle\cdot\rangle$ denotes expectation over q. This is equivalent to minimizing the Kullback–Leibler divergence $D_{\mathrm{KL}}(q\|p)$ between q and p [1, 5].

Finding the optimal approximation can be seen as an optimization problem, where the lower bound $\mathcal{B}(q(\boldsymbol{\theta}|\boldsymbol{\xi}))$ is maximized with respect to the variational parameters $\boldsymbol{\xi}$. This is often solved using a VB EM algorithm by updating sets of parameters alternatively while keeping the others fixed. Both VB-E and VB-M steps can implicitly optimally utilize the Riemannian structure of $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ for conjugate exponential family models [10]. Nevertheless, the EM based methods are prone to slow convergence, especially under low noise, even though more elaborate optimization schemes can speed up their convergence somewhat.

The formulation of VB as an optimization problem allows applying generic optimization algorithms to maximize $\mathcal{B}(q(\boldsymbol{\theta}|\boldsymbol{\xi}))$, but this is rarely done in practice because the problems are quite high dimensional. Additionally other parameters may influence the effect of other parameters and the lack of this specific knowledge of the geometry of the problem can seriously hinder generic optimization tools.

Assuming the approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ is Gaussian, it is often enough to use generic optimization tools to update the mean of the distribution. This is because the negative entropy of a Gaussian $q(\boldsymbol{\theta}|\boldsymbol{\mu},\boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

is $\langle \log q(\boldsymbol{\theta}|\boldsymbol{\xi}) \rangle = -\frac{1}{2} \log \det(2\pi e \boldsymbol{\Sigma})$ and thus straightforward differentiation of Eq. (1) yields a fixed point update rule for the covariance

$$\boldsymbol{\Sigma}^{-1} = -2\nabla_{\boldsymbol{\Sigma}} \langle \log p(\boldsymbol{X}, \boldsymbol{\theta}) \rangle . \tag{2}$$

If the covariance is assumed diagonal, the same update rule applies for the diagonal terms.

3 Natural gradient learning for VB

Let $\mathcal{F}(\boldsymbol{\xi})$ be a scalar function defined on the manifold $S = \{\boldsymbol{\xi} \in \mathbf{R}^n\}$. If S is a Euclidean space and the coordinate system $\boldsymbol{\xi}$ is orthonormal, the direction of steepest ascent is given by the standard gradient $\nabla \mathcal{F}(\boldsymbol{\xi})$.

If the space S is a curved Riemannian manifold, the direction of steepest ascent is given by the natural gradient [9]

$$\tilde{\nabla} \mathcal{F}(\boldsymbol{\xi}) = \mathbf{G}^{-1}(\boldsymbol{\xi}) \nabla \mathcal{F}(\boldsymbol{\xi}). \tag{3}$$

The $n \times n$ matrix $\mathbf{G}(\boldsymbol{\xi}) = (g_{ij}(\boldsymbol{\xi}))$ is called the Riemannian metric tensor and it may depend on the point of origin $\boldsymbol{\xi}$.

For the space of probability distributions $q(\boldsymbol{\theta}|\boldsymbol{\xi})$, the most common Riemannian metric tensor is given by the Fisher information [8]

$$I_{ij}(\boldsymbol{\xi}) = g_{ij}(\boldsymbol{\xi}) = E\left\{ \frac{\partial \ln q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_i} \frac{\partial \ln q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_j} \right\} = E\left\{ -\frac{\partial^2 \ln q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_i \partial \xi_j} \right\}, \tag{4}$$

where the last equality is valid given certain regularity conditions [11].

3.1 Computing the Riemannian metric tensor

When applying natural gradients to approximate inference, the geometry is defined by the approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ and not the full model $p(\boldsymbol{X}|\boldsymbol{\theta})$ as usually. If the approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ is chosen such that disjoint groups of variables are independent, that is,

$$q(\boldsymbol{\theta}|\boldsymbol{\xi}) = \prod_i q_i(\boldsymbol{\theta}_i|\boldsymbol{\xi}_i), \tag{5}$$

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

$$\operatorname{diag}(A_1, \ldots, A_n)^{-1} = \operatorname{diag}(A_1^{-1}, \ldots, A_n^{-1}). \tag{6}$$

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

Fig. 1. 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) [12].

3.2 Gaussian distribution

For the univariate Gaussian distribution parametrized by mean and variance $N(x;\ \mu, v)$, we have

$$\ln q(x|\mu, v) = -\frac{1}{2v}(x-\mu)^2 - \frac{1}{2}\ln(v) - \frac{1}{2}\ln(2\pi). \tag{7}$$

Furthermore,

$$E\left\{-\frac{\partial^2 \ln q(x|\mu, v)}{\partial\mu\partial\mu}\right\} = \frac{1}{v}, \tag{8}$$

$$E\left\{-\frac{\partial^2 \ln q(x|\mu, v)}{\partial v\partial\mu}\right\} = 0, \text{ and} \tag{9}$$

$$E\left\{-\frac{\partial^2 \ln q(x|\mu, v)}{\partial v\partial v}\right\} = \frac{1}{2v^2}. \tag{10}$$

The vanishing of the cross term between mean and variance further supports using the simpler fixed point rule (2) to update the variances.

In the case of univariate Gaussian distribution, natural gradient for the mean has a rather straightforward intuitive interpretation, which is illustrated in Figure 1 (left). Compared to conventional gradient, natural 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, the elements of the Fisher information matrix corresponding to the mean are simply

$$E\left\{-\frac{\partial^2 \ln q(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial\boldsymbol{\mu}^T\partial\boldsymbol{\mu}}\right\} = \boldsymbol{\Sigma}^{-1}. \tag{11}$$

Typically the covariance matrix $\boldsymbol{\Sigma}$ is assumed to have a simple structure (diagonal, diagonal+rank-k, simple Markov random field) that makes working with it very efficient.

4 Natural and conjugate gradient methods

Many of the traditional optimization algorithms have their direct counterparts in Riemannian space. This paper concentrates on gradient based algorithms, in particular the generalized versions of gradient ascent and conjugate gradient method.

Gradient-based optimization algorithms in Euclidean space operate by deriving a search direction using the gradient at current search point and possibly other information. Then, either a fixed-length step is taken or a line search performed in this direction. The fixed step length can still be adjusted during learning.

When generalizing these methods to Riemannian space, the geometrically most natural approach would be to take the steps or perform the line search along geodesics, which are length-minimizing curves and hence Riemannian counterparts of straight lines. In practice this is rarely done because the mathematical forms of geodesics can be very complicated thus making operations with them computationally expensive. Euclidean straight lines are used instead of geodesics in this work as well.

4.1 Natural gradient ascent

The natural gradient learning algorithm is analogous to conventional gradient ascent algorithm and is given by the iteration

$$\boldsymbol{\xi}_k = \boldsymbol{\xi}_{k-1} + \gamma \tilde{\nabla} \mathcal{F}(\boldsymbol{\xi}_{k-1}), \tag{12}$$

where the step size γ can either be adjusted adaptively during learning [9] or computed for each iteration using e.g. line search. In general, the performance of natural gradient learning is superior to conventional gradient learning when the problem space is Riemannian; see [9].

4.2 Conjugate gradient methods and Riemannian conjugate gradient

For better performance it can be useful to combine natural gradient learning with some standard superlinear optimization algorithm. One such algorithm is the nonlinear conjugate gradient (CG) method [13]. The conjugate gradient method is a standard tool for solving high dimensional nonlinear optimization problems. During each iteration of the conjugate gradient method, a new search direction is generated by conjugation of the residuals from previous iterations. With this choice the search directions form a Krylov subspace and only the previous search direction and the current gradient are required for the conjugation process, making the algorithm efficient in both time and space complexity [13].

The extension of the conjugate gradient algorithm to Riemannian manifolds is done by replacing the gradient with the natural gradient. The resulting algorithm is known as the Riemannian conjugate gradient method [14, 15]. In

principle this extension is relatively simple, as it is sufficient that all the vector operations take into account the Riemannian nature of the problem space. Therefore, the line searches are performed along geodesic curves and the old gradient vectors $\tilde{\mathbf{g}}_{k-1}$ defined in a different tangent space are transformed to the tangent space at the origin of the new gradient by parallel transport along a geodesic [14].

4.3 Natural conjugate gradient

Like with natural gradient ascent, it is often necessary to make certain simplifying assumptions to keep the iteration simple and efficient. In this paper, the geodesic curves used in Riemannian conjugate gradient algorithm are approximated with (Euclidean) straight lines. This also means that parallel transport cannot be used, and vector operations between vectors from two different tangent spaces are performed in the Euclidean sense, i.e. assuming that the parallel transport between two points close to each other on the manifold can be approximated by the identity mapping. This approximative algorithm is called the natural conjugate gradient (NCG).

For small step sizes and geometries which are locally close to Euclidean these assumptions still retain many of the benefits of original algorithm while greatly simplifying the computations. Edelman et al. [15] 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.

The search direction for the natural conjugate gradient method is given by

$$\mathbf{p}_k = \tilde{\mathbf{g}}_k + \beta \mathbf{p}_{k-1}, \tag{13}$$

and the Polak-Ribiére formula used to evaluate the coefficient β is given by

$$\beta = \frac{(\tilde{\mathbf{g}}_k - \tilde{\mathbf{g}}_{k-1}) \cdot \tilde{\mathbf{g}}_k}{\tilde{\mathbf{g}}_{k-1} \cdot \tilde{\mathbf{g}}_k}. \tag{14}$$

5 VB for nonlinear state-space models

As a specific example, we consider the nonlinear state-space model (NSSM) introduced in [5]. The model is specified by the generative model

$$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) + \mathbf{n}(t) \tag{15}$$

$$\mathbf{s}(t) = \mathbf{s}(t-1) + \mathbf{g}(\mathbf{s}(t-1), \boldsymbol{\theta}_\mathbf{g}) + \mathbf{m}(t), \tag{16}$$

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 modeled with multilayer perceptron (MLP) networks. Observation noise $\mathbf{n}$ and process noise $\mathbf{m}$ are assumed Gaussian. 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. The posterior approximation of the states $q(\boldsymbol{\theta}_{\boldsymbol{S}}|\boldsymbol{\xi}_{\boldsymbol{S}})$ is a Gaussian Markov random field a correlation between the corresponding components of subsequent state vectors $s_j(t)$ and $s_j(t-1)$. This is a realistic minimum assumption for modeling the dependence of the state vectors $\mathbf{s}(t)$ and $\mathbf{s}(t-1)$ [5].

Because of the nonlinearities the model is not in the conjugate exponential family, and the standard VB learning methods are only applicable to hyperparameters and not the latent states or weights of the MLPs. The bound (1) can nevertheless be evaluated by linearizing the MLP networks $\mathbf{f}$ and $\mathbf{g}$ using the technique of [7]. 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 optimizer to adapt the parameters. The natural gradient for the mean elements is given by

$$\tilde{\nabla}_{\boldsymbol{\mu}_q}\mathcal{F}(\boldsymbol{\xi}) = \boldsymbol{\Sigma}_q \nabla_{\boldsymbol{\mu}_q}\mathcal{F}(\boldsymbol{\xi}), \tag{17}$$

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 of the model parameters is diagonal while the inverse covariance 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 were updated separately using a fixed-point update rule similar to (2) as described in [5].

6 Experiments

As an example, the method for learning nonlinear state-space models presented in Sec. 5 was applied to real world speech data. Experiments were made 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.

To study the performance differences between the natural conjugate gradient (NCG) method, standard natural gradient (NG) method, standard conjugate gradient (CG) method and the heuristic algorithm from [5], 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 the E-step failed due to instability of the used Kalman filtering algorithm.

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 mapping had 20 hidden nodes. Four different initializations 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 iteration was assumed to have converged when $|\mathcal{B}^t - \mathcal{B}^{t-1}| < \varepsilon = (10^{-5}N/500)$ for 5 consecutive iterations, where $\mathcal{B}^t$ is the bound on marginal log-likelihood at iteration t and N is the size of the data set. Alternatively, the

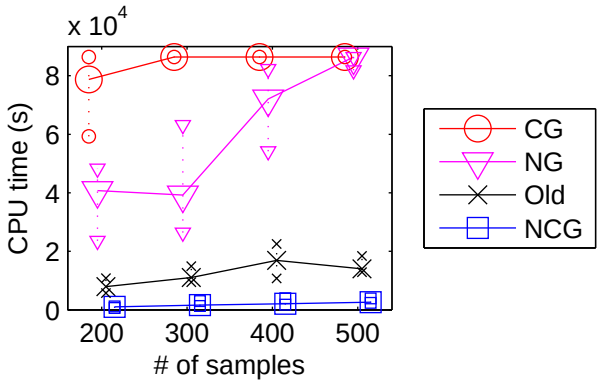

Fig. 2. Convergence speed of the natural conjugate gradient (NCG), the natural gradient (NG) and the conjugate gradient (CG) methods as well as the heuristic algorithm (Old) with different data sizes. The lines show median times with 25 % and 75 % quantiles shown by the smaller marks.

iteration was stopped after 24 hours even if it had not converged. Practically all the simulations converged to different local optima, but there were no statistically significant differences in the bound values corresponding to these optima (Wilcoxon rank-sum test, 5 % significance level). There were still some differences, and especially the NG 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 bound value that was more than two NCG standard deviations worse than NCG average for the particular data set. The results of one run where the heuristic algorithm diverged were also discarded from the analysis.

The results can be seen in Figure 2. The plain CG and NG methods were clearly slower than others and the maximum runtime was reached by most CG and some NG runs. NCG was clearly the fastest algorithm with the heuristic method between these extremes.

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 NCG, NG, CG methods and the heuristic algorithm were compared. The results can be seen in Figure 3. The results show the convergence for five different initializations with markers at the end showing when the convergence was reached.

NCG clearly outperformed the other algorithms in this experiment as well. In particular, both NG and CG hit the maximum runtime in every run, and especially CG was nowhere near convergence at this time. NCG also outperformed the heuristic algorithm [5] by a factor of more than 10.

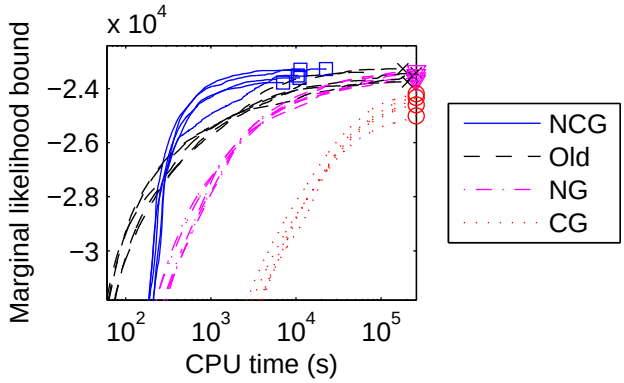

Fig. 3. Comparison of the performance of the natural conjugate gradient (NCG), the natural gradient (NG), the conjugate gradient (CG) methods and the heuristic algorithm with the full data set. Lower bound on marginal log-likelihood $\mathcal{B}$ is plotted against computation time.

7 Discussion

In previous machine learning algorithms based on natural gradients [9], 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 for data $p(\boldsymbol{X}|\boldsymbol{\theta})$. The resulting geometry is often much more complicated as the effects of different parameters cannot be separated and the Fisher information matrix is relatively dense. In this paper, only the simpler geometry of the approximating distributions $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ is used. Because the approximations are often chosen to minimize 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 invert.

While taking into account the structure of the approximation, plain natural gradient in this case ignores the structure of the model and the global geometry of the parameters $\boldsymbol{\theta}$. This is to some extent addressed by using conjugate gradients, and even more sophisticated optimization methods such as quasi-Newton or even Gauss–Newton methods can be used if the size of the problem permits it.

While the natural conjugate gradient method has been formulated mainly for models outside the conjugate-exponential family, it can also be applied to conjugate-exponential models instead of the more common VB EM algorithms. In practice, simpler and more straightforward EM acceleration methods may still provide comparable results with less human effort.

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 too strong for model variables with small posterior variance

and too weak for variables with large posterior variance, as seen from Eqs. (8)–(10). The posterior variance of latent variables is often much larger than the posterior variance of model parameters. The difference grows with the size of the data set: When the data set is small, regular conjugate gradient method works somehow, but for larger data sets, natural conjugate gradient shows far superior performance.

Acknowledgments This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

1. Bishop, C.: Pattern Recognition and Machince Learning. Springer, Cambridge (2006)
2. Barber, D., Bishop, C.: Ensemble learning for multi-layer networks. In Advances in Neural Information Processing Systems 10. The MIT Press, Cambridge, MA, USA (1998) 395–401
3. Seeger, M.: Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers. In Advances in Neural Information Processing Systems 12. MIT Press, Cambridge, MA, USA (2000) 603–609
4. Lappalainen, H., Honkela, A.: Bayesian nonlinear independent component analysis by multi-layer perceptrons. In Girolami, M., ed.: Advances in Independent Component Analysis. Springer-Verlag, Berlin (2000) 93–121
5. Valpola, H., Karhunen, J.: An unsupervised ensemble learning method for nonlinear dynamic state-space models. Neural Computation **14**(11) (2002) 2647–2692
6. Valpola, H., Harva, M., Karhunen, J.: Hierarchical models of variance sources. Signal Processing **84**(2) (2004) 267–282
7. Honkela, A., Valpola, H.: Unsupervised variational Bayesian learning of nonlinear models. In Advances in Neural Information Processing Systems 17. MIT Press, Cambridge, MA, USA (2005) 593–600
8. Amari, S.: Differential-Geometrical Methods in Statistics. Volume 28 of Lecture Notes in Statistics. Springer-Verlag (1985)
9. Amari, S.: Natural gradient works efficiently in learning. Neural Computation **10**(2) (1998) 251–276
10. Sato, M.: Online model selection based on the variational Bayes. Neural Computation **13**(7) (2001) 1649–1681
11. Murray, M.K., Rice, J.W.: Differential Geometry and Statistics. Chapman & Hall (1993)
12. Valpola, H.: 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.
13. Nocedal, J.: Theory of algorithms for unconstrained optimization. Acta Numerica **1** (1991) 199–242
14. Smith, S.T.: Geometric Optimization Methods for Adaptive Filtering. PhD thesis, Harvard University, Cambridge, Massachusetts (1993)
15. Edelman, A., Arias, T.A., Smith, S.T.: The geometry of algorithms with orthogonality constraints. SIAM Journal on Matrix Analysis and Applications **20**(2) (1998) 303–353