Research Collaboration Report

A three month visit to Princeton University supported by Marshallplan Scholarship

Thomas Buchgraber[1]

November 2, 2010

Abstract

In this document, the outcome of the research collaboration of Thomas Buchgraber at the University of Princeton, Department of Electrical Engineering is presented. He was working in the group of Prof. H. Vincent Poor from May 11, 2010 until August 12, 2010. The work was focused on sparse variational inference planned for the use in online distributed learning algorithms for applications in wireless sensor networks.

1 Introduction

To motivate the research on sparse signal representation in the field of distributed learning algorithms, we start this section with a general introduction on wireless sensor networks (WSNs). In the following we will focus on sparse representation and in particular on sparse Bayesian learning (SBL), where we point out why such methods are essentially usefull for WSNs. We discuss an approximate inference method called variational inference at the end of this section to lay the basis for further discussions. Variational inference provides a method to update unknown model parameters in a space alternating way and makes SBL inference even possible, since the joint posterior of all unknown model parameters is not tractable in closed form.

The rest of the report is organized as follows. In Section 2 we introduce variational SBL, a rather slow method that we wish to accelerate. We do so in Section 3, where we describe the work that has been done during the collaboration at Princeton which was focused on fast variational SBL. Section 4 finally presents the publications that have been made based on the research collaboration and also points out the ongoing and future directions of this research.

1.1 Wireless Sensor Networks

WSNs have gained tremendous attention during the last years [1] and consist of sensor devices capable to communicate with each other. They are usually deployed randomly in a region under

[1]The author wants to thank the Austrian Marshall Plan Foundation for supporting this research. Also the author would like to thank Prof. Vincent Poor for providing the facilities and for making this collaboration happen. Special thanks goes to Dr. Dmitriy Shutin for many insightfull comments, discussions and the close collaboration in general.

scrutiny and have tight energy and bandwith constraints such that the amount of data that can be transmitted and local computations that can be carried out is limited. Designed to make inferences about the environment which they are sensing, a major goal is to distribute the computational effort of the used algorithms among the nodes to save energy and reduce the communicational load of the individual sensors. This is opposed to a centralized approach, where the measurements have to be transmitted to a fusion-center (powerful central base station) that then carries out the computations. Since many real world spatial-temporal phenomena (e.g. air pressure- or temperature-fields) tend to be very complex, transmission of the sensor information to a fusion-center is impractical in terms of the whole networks lifetime. This is due to the limited transmission range of each sensing-device, such that many nodes need to relay information from distant parts of the network to the fusion center.

1.2 Sparse Representation and Sparse Bayesian Learning

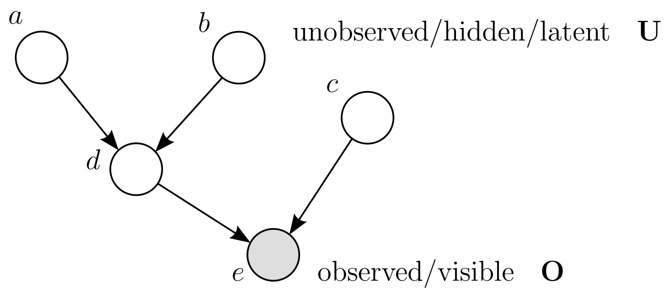
In the previous section we discussed the energy and computational limitations of sensor nodes in a WSN. To further reduce the communication load of a distributed algorithm we only need to communicate the essential information needed for processing, i.e. we should find a sparse representation. One sparsification method based on probabilistic models is SBL, where we can automatically learn both, which of the parameters are important and which values they should have. But before we discuss the details of SBL we start with a general introduction on supervised learning, a concept in machine learning.

In supervised learning, one is given a set of input-output pairs $\{\boldsymbol{x}_i, t_i\}_{i=1}^{N}$, where the input usually is a d-dimensional real vector $\boldsymbol{x} \in \mathbb{R}^d$ and the desired model output or target is a real valued scalar $t \in \mathbb{R}$. Learning in this sense means to find the function that links the inputs to the targets bases on the training examples. To do this we need a model for the targets. Because of mathematical convenience it is very common to use a linear combination of potentially non-linear basis functions to model the targets, i.e.

$$\boldsymbol{t} = \boldsymbol{\Phi}\boldsymbol{w} + \boldsymbol{\epsilon} \, , \tag{1}$$

where $\boldsymbol{\Phi} = [\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_L]$ is a design matrix consisting of L basis vectors $\boldsymbol{\varphi}_l = [\psi_l(\boldsymbol{x}_1), \ldots, \psi_l(\boldsymbol{x}_N)]^T$ with basis functions $\psi_l(\cdot)$, $\boldsymbol{t} = [t_1, \ldots, t_N]^T$ is a vector of targets, $\boldsymbol{w} \in \mathbb{R}^L$ a weight vector and $\boldsymbol{\epsilon}$ a zero mean additive white Gaussian perturbation vector with covariance matrix $\tau^{-1}\mathbf{I}$. In sparse signal representation, the aim is to find a small number of non-zero weights in $\boldsymbol{w}$ to represent the targets $\boldsymbol{t}$, that means we are not only interessted in the values but also which of the weight parameters are relevant for the model.

In SBL [3], [10], a probabilistic model for obtaining sparsity is used. We define the probability density functions (pdfs) in the following. An observation likelihood function $p(\boldsymbol{t}|\boldsymbol{w}, \tau) =$



a
b
unobserved/hidden/latent $\mathbf{U}$
c
d
e
observed/visible $\mathbf{O}$

Figure 1: A BN in form of a DAG

$\mathcal{N}(\boldsymbol{t}|\boldsymbol{\Phi}\boldsymbol{w}, \tau^{-1}\mathbf{I})$ is given from (1) and additionally we define a hierarchical weight prior $p(\boldsymbol{w}|\boldsymbol{\alpha}) = \prod_{l=1}^{L} \mathcal{N}(w_l|0, \alpha_l^{-1})$ which lays the foundation of SBL. Using a common definition of the Gamma distribution $\text{Ga}(x|a,b) = \frac{b^a}{\Gamma(a)} x^{a-1} \exp(-bx)$, the remaining prior pdfs are given by $p(\alpha_l) = \text{Ga}(\alpha_l|a_l, b_l)$ and $p(\tau) = \text{Ga}(\tau|c,d)$. By setting the prior parameters $a_l = b_l = 0, \forall l$, which renders the prior as non-informative, automatic relevance determination (ARD) is achieved. Using Bayes' rule we are interessted in the posterior pdf of all unknown variables given the data

$$p(\boldsymbol{w}, \boldsymbol{\alpha}, \tau|\boldsymbol{t}) = \frac{p(\boldsymbol{t}|\boldsymbol{w}, \tau)p(\boldsymbol{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})p(\tau)}{p(\boldsymbol{t})}, \tag{2}$$

where we have used $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_L]^T$. Unfortunately the marginal $p(\boldsymbol{t})$ in the denominator of (2) can not be computed in closed form and thus the posterior $p(\boldsymbol{w}, \boldsymbol{\alpha}, \tau|\boldsymbol{t})$ is intractable [10]. To find a solution for the SBL problem we thus must seek an approximation to the true posterior. This can be done by using variational inference which will be introduced in the next section.

1.3 Variational Inference

Consider an example Bayesian network (BN) depicted in Figure 1. Random variables are represented by nodes, where shaded ones are considered as observed and unshaded as hidden[2] random variables. For this section we denote the set of all hidden variables as $\mathbf{U}$ and the set of observed variables as $\mathbf{O}$. A BN is a graphical representation of the joint probability of random variables by showing the conditions between them. The tails of the arrows heading to a node depict the variables which the headed node is conditioned on. The example in Figure 1 represents a joint probability of

$$p(a,b,c,d,e) = p(e|d,c)\,p(d|a,b)\,p(a)\,p(b)\,p(c),$$

where nodes with no incomming arrows depict priors. Because there are no loops in the presented graph, the example illustrates a special case of a BN called directed acyclic graph (DAG).

[2]The synonym unobserved or latent is often also used for the hidden random variables and similarly the term visible refers to the observed variables.

For further information on BNs we refer the reader to [2].

With any approximating distribution $q(\mathbf{U})$ we can write the log-marginal distribution of the set of observed variables as

$$\ln p(\mathbf{O}) = \int q(\mathbf{U})\, d\mathbf{U}\ \ln p(\mathbf{O}), \tag{3}$$

with $\int q(\mathbf{U})\, d\mathbf{U} = 1$, because we assume that $q(\mathbf{U})$ is a valid normalized probaility distribution function (pdf)[3] and thus should always integrate to one. We can now plug the log-marginal inside the integral

$$\ln p(\mathbf{O}) = \int q(\mathbf{U})\ \ln p(\mathbf{O})\, d\mathbf{U}$$

since it is not a function of $\mathbf{U}$. By multiplying the inside term of the logarithm with terms that give one, thus we do not change the result of the equation, we bring in the posterior of the hidden variables given the observations $p(\mathbf{U}|\mathbf{O})$ and the proxy distribution $q(\mathbf{U})$ over the latent variables, i.e.

$$\begin{aligned}\ln p(\mathbf{O}) &= \int q(\mathbf{U})\ \ln\left\{\frac{p(\mathbf{O})p(\mathbf{U}|\mathbf{O})q(\mathbf{U})}{p(\mathbf{U}|\mathbf{O})q(\mathbf{U})}\right\} d\mathbf{U}\\ &= \int q(\mathbf{U})\ \ln\left\{\frac{p(\mathbf{O},\mathbf{U})\cdot q(\mathbf{U})}{q(\mathbf{U})\cdot p(\mathbf{U}|\mathbf{O})}\right\} d\mathbf{U}.\end{aligned}$$

Using the rule that the log of a product is the sum of the logs separates the terms as

$$\ln p(\mathbf{O}) = \underbrace{\int q(\mathbf{U})\ \ln\left\{\frac{p(\mathbf{O},\mathbf{U})}{q(\mathbf{U})}\right\} d\mathbf{U}}_{\mathcal{L}(q)} + \underbrace{\int q(\mathbf{U})\ \ln\left\{\frac{q(\mathbf{U})}{p(\mathbf{U}|\mathbf{O})}\right\} d\mathbf{U}}_{KL(q||p)} \tag{4}$$

and results in a sum of $\mathcal{L}(q)$, the variational lower bound and $KL(q||p)$, a Kullback-Leibler divergence. The aim of variational inference is to find an approximation $q(\mathbf{U})$ of the posterior $p(\mathbf{U}|\mathbf{O})$. Since the log-marginal on the left hand side of (4) is not a functional[4] of $q(\mathbf{U})$, it stays unchanged with respect to changes of the proxy $q(\mathbf{U})$. Because by definition a Kullback-Leibler divergence is always positive, $KL(q||p)$ is only zero for the case $q(\mathbf{U}) = p(\mathbf{U}|\mathbf{O})$. We thus see that $\mathcal{L}(q)$ must be a lower bound of the log-marginal $\ln p(\mathbf{O})$, where the bound is only achieved if the proxy pdf perfectly reaches the posterior pdf. Thus, maximizing the variational lower bound is equivalent to minimizing the Kullback-Leibler divergence $KL(q||p)$ as a functional of $q(\mathbf{U})$ that uses the joint pdf instead of the usually intractable posterior pdf.

The proxy pdf $q(\mathbf{U})$ is usually assumed to be of a factorized structure, which is refered to

[3]Not that we talk about continuous random variables here. All the derivations are also valid for discrete random variables, where in this case we must change the integral in (3) to a summation.

[4]A functional is a function of a function. Like e.g. the variational lower bound $\mathcal{L}(q)$ is a function of the proxy pdf $q(\mathbf{U})$, which itself is a function.

as mean field approximation [2] and is given by

$$q(\mathbf{U}) = \prod_{i=1}^{M} q_i(\mathbf{U}_i) = \prod_i q_i, \tag{5}$$

where we have introduced the short form notation q_i for the term $q_i(\mathbf{U}_i)$ for all M factors. The variational lower bound can now be analysed as a functional of just one factor q_j in (5)

$$\begin{aligned} \mathcal{L}(q_j) &= \int \left\{ \ln p(\mathbf{O}, \mathbf{U}) - \ln q_j - \sum_{i \neq j} \ln q_i \right\} q_j \, d\mathbf{U}_j \prod_{i \neq j} q_i \, d\mathbf{U}_i \\ &= \int q_j \Bigg\{ \underbrace{\int \ln p(\mathbf{O}, \mathbf{U}) \prod_{i \neq j} q_i \, d\mathbf{U}_i}_{\mathbb{E}_{i \neq j}[\ln p(\mathbf{O},\mathbf{U})]} - \ln q_j \Bigg\} d\mathbf{U}_j + \text{const.} \end{aligned} \tag{6}$$

and we use the notation “const.” to denote all the terms that are not depending on q_j. The term $\mathbb{E}_{i \neq j}[\cdot]$ means an expectation with respect to all $M-1$ proxy factors q_i for all $i \neq j$. We can now introduce

$$\ln \tilde{p}_j = \mathbb{E}_{i \neq j}\left[\ln p(\mathbf{O}, \mathbf{U})\right] + \text{const.} \quad \longrightarrow \quad \tilde{p}_j \propto \exp\Big\{ \mathbb{E}_{i \neq j}\left[\ln p(\mathbf{O}, \mathbf{U})\right] \Big\}, \tag{7}$$

a normalized distribution $\tilde{p}_j$ with “const.” equal to the normalization of the pdf. It is easy to see from (6) that by using definition (7) we obtain

$$\mathcal{L}(q_j) = -KL\left(q_j || \tilde{p}_j\right) + \text{const.}, \tag{8}$$

a Kullback-Leibler divergence between the proxy q_j and the newly defined $\tilde{p}_j$. With this result and the knowlege that the Kullback-Leibler divergence is at its minimum if $q_j = \tilde{p}_j$ we can find the maximum of the variational lower bound with respect to one of the factors in (5) as $q_j^* = \tilde{p}_j$. If the proxy factor q_j is constraint to be from a class of density functions $\mathcal{Q}_j$, then the optimal density is generally defined as

$$q_j^* = \underset{q_j \in \mathcal{Q}_j}{\operatorname{argmin}} \Big\{ KL\left(q_j || \tilde{p}_j\right) \Big\}. \tag{9}$$

For instance, if $\mathcal{Q}_j$ is the space of Gaussian densities, then we can use the *Laplace approximation method* to find the mean and covariance of $q_j^*(\mathbf{Z}_j)$. Thus, we just compute the sufficient statistics of the Gaussian.

2 Variational Sparse Bayesian Learning

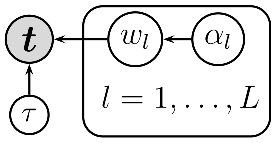


Figure 2: A DAG representing the SBL problem. Gray nodes represent observed variables.

In variational sparse Bayesian learning, exemplified by the variational relevance vector machine [3], a directed acyclic graph (DAG) presented in Figure 2 is considered. This graph represents the general SBL problem. The joint pdf given by the graph factorizes as

$$p(\boldsymbol{w}, \boldsymbol{t}, \boldsymbol{\alpha}, \tau) = p(\boldsymbol{t}|\boldsymbol{w}, \tau)p(\boldsymbol{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})p(\tau). \tag{10}$$

Like has been discussed in Section 1.2, the derivation of the posterior $p(\boldsymbol{w}, \boldsymbol{\alpha}, \tau|\boldsymbol{t})$ is intractable and variational inference tries to approximate it by using a proxy pdf $q(\boldsymbol{w}, \boldsymbol{\alpha}, \tau)$ that maximizes the variational lower bound $\mathcal{L}(q(\boldsymbol{w}, \boldsymbol{\alpha}, \tau))$

$$\begin{aligned} \ln p(\boldsymbol{t}) \;&\geq \int q(\boldsymbol{w}, \boldsymbol{\alpha}, \tau) \ln \frac{p(\boldsymbol{w}, \boldsymbol{t}, \boldsymbol{\alpha}, \tau)}{q(\boldsymbol{w}, \boldsymbol{\alpha}, \tau)} d\boldsymbol{w} d\boldsymbol{\alpha} d\tau \\ &= \mathcal{L}(q(\boldsymbol{w}, \boldsymbol{\alpha}, \tau)), \end{aligned} \tag{11}$$

which is a lower bound for the log-evidence $\ln p(\boldsymbol{t})$. Applying mean field approximation, in [3] it is assumed that $q(\boldsymbol{w}, \boldsymbol{\alpha}, \tau)$ is factored as

$$q(\boldsymbol{w}, \boldsymbol{\alpha}, \tau) = q(\boldsymbol{w})q(\tau) \prod_{l=1}^{L} q(\alpha_l) \tag{12}$$

and the individual pdfs are defined as follows: $q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$, $q(\alpha_l) = \mathrm{Ga}(\alpha_l|\hat{a}_l, \hat{b}_l)$ and $q(\tau) = \mathrm{Ga}(\tau|\hat{c}, \hat{d})$. Maximization of the lower bound (11) with respect to the individual factors in (12) gives the following update equations [3]:

$$\hat{\boldsymbol{\Sigma}} = \left(\hat{\tau}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \mathrm{diag}(\hat{\boldsymbol{\alpha}})\right)^{-1} \tag{13}$$

$$\hat{\boldsymbol{\mu}} = \hat{\tau}\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{t} \tag{14}$$

$$\hat{a}_l = a_l + 1/2, \quad \hat{b}_l = b_l + (\hat{\mu}_l{}^2 + \hat{\Sigma}_{ll})/2 \tag{15}$$

$$\hat{c} = c + \frac{N}{2}, \quad \hat{d} = d + \frac{||\boldsymbol{t} - \boldsymbol{\Phi}\hat{\boldsymbol{w}}||^2 + \mathrm{tr}(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{\Phi})}{2}, \tag{16}$$

where $\hat{\tau} = \mathbb{E}_{q(\tau)}\{\tau\} = \hat{c}/\hat{d}$, $\hat{\boldsymbol{\alpha}} = [\hat{\alpha}_1, \ldots, \hat{\alpha}_L]^T$, $\hat{\alpha}_l = \mathbb{E}_{q(\alpha_l)}\{\alpha_l\} = \hat{a}_l/\hat{b}_l$, $\hat{\mu}_l$ is the lth element of the vector $\hat{\boldsymbol{\mu}}$ and $\hat{\Sigma}_{ll}$ is the lth main diagonal element of the matrix $\hat{\boldsymbol{\Sigma}}$. Note that by $\mathbb{E}_{q(z)}\{\cdot\}$ we mean the expectation with respect to the distribution $q(z)$ here. To find the sufficient statistics of the proxy pdf $q(\boldsymbol{w}, \boldsymbol{\alpha}, \tau)$, one has to iterate through the terms given in (13)-(16) until the variational lower bound (11) converges to a maximum. Because the bound is convex [2] with respect to each of the factors given in (12) we can update them in any arbitrary order.

3 Research at Princeton University

In this section we present the results achieved during the research collaboration at Princeton University, Department of Electrical Engineering. The work was performed in close collaboration with Dr. Dmitriy Shutin. In Section 3.1 we show how the rather slow variational SBL method presented in Section 2 can be performed much faster using a fixed point method. Then in Section 3.2 we start a discussion of how the fast method can be implemented efficiently by only using rank-1 updates. Finally, in Section 3.3 we show how the new proposed method performs for different simulation settings and how the derived pruning condition can be modified to achieve better results for some situations.

3.1 Fast Variational Sparse Bayesian Learning

Essentially, the variational update expressions (13)-(16) provide the estimates of the moments of the corresponding approximating pdfs. Although these expressions reduce to those obtained in [10] when the approximating factors $q(\tau)$, and $q(\alpha_l)$ are chosen as Dirac measures on the corresponding domains[5], they do not reveal the structure of the marginal likelihood function that leads to the fast implementation of SBL.

Consider now the variational update expression (13)-(15). Due to the convexity of the variational lower bound functional in the approximating factors $q(\boldsymbol{w})$, $q(\tau)$, $q(\alpha_l)$, $l = 1, \ldots, L$ [2], we can update these factors in any order; furthermore, a group of factors can be updated successively while keeping the other factors fixed. In [11] the authors investigate the dependence of the marginal likelihood on a single sparsity parameter $\hat{\alpha}_l$; it is this dependence that leads to an efficient implementation of SBL. Thus it makes sense to inquire into a fixed point of the variational update expression for a single factor $q(\alpha_l)$.

Let us now select the ARD case $a_l = b_l = 0$ for all the hyperpriors $p(\alpha_l)$ and consider the expression for the mean $\hat{\alpha}_l$ of $q(\alpha_l)$ for some fixed l.[6] From (15) and the properties of a Gamma

[5]In [10] the posterior pdf of the weights $\boldsymbol{w}$ is Gaussian; its parameters coincide with the variational parameters of $q(\boldsymbol{w})$ in (13) and (14).

[6]Notice that since $a_l = b_l = 0$, the parameters of $q(\alpha_l)$ in (15) are given by $\hat{a}_l = 1/2$ and $\hat{b}_l = 1/(2\hat{\alpha}_l)$. Thus it makes sense to study the fixed point of the variational update expression in terms of $\hat{\alpha}_l$, rather than in terms of $\hat{a}_l$ and $\hat{b}_l$.

distribution it follows that

$$\hat{\alpha}_l^{-1} = \boldsymbol{e}_l^T(\hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^T + \hat{\boldsymbol{\Sigma}})\boldsymbol{e}_l, \tag{17}$$

where $\boldsymbol{e}_l = [0, \dots, 0, 1, 0, \dots, 0]^T$ is a vector of all zeros with 1 at the lth position. The variational parameters $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ of $q(\boldsymbol{w})$ both implicitly depend on $\hat{\alpha}_l$. Let us now assume that $q(\boldsymbol{w})$ and $q(\alpha_l)$ are successively updated while keeping $q(\tau)$ fixed. This will generate a sequence of estimates $\{\hat{\alpha}_l^{[m]}\}_{m=1}^M$, with each element in the sequence computed according to (17). Our goal is to compute the fixed point $\hat{\alpha}_l^{[\infty]}$ of this sequence as $M \to \infty$.

First, we define $\mathbf{B} = \hat{\tau}^2\boldsymbol{\Phi}^T\boldsymbol{t}\boldsymbol{t}^T\boldsymbol{\Phi}$ it is easy to see that

$$\hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^T = \hat{\boldsymbol{\Sigma}}\mathbf{B}\hat{\boldsymbol{\Sigma}}^T. \tag{18}$$

Now consider the influence of a single sparsity parameter $\hat{\alpha}_l$ on the matrix $\hat{\boldsymbol{\Sigma}}$ in (13). By noting that $\text{diag}(\hat{\boldsymbol{\alpha}}) = \sum_j \hat{\alpha}_j\boldsymbol{e}_j\boldsymbol{e}_j^T$ we rewrite $\hat{\boldsymbol{\Sigma}}$ as

$$\begin{aligned}\hat{\boldsymbol{\Sigma}} &= \left(\hat{\tau}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \sum_{j\neq l}\hat{\alpha}_j\boldsymbol{e}_j\boldsymbol{e}_j^T + \hat{\alpha}_l\boldsymbol{e}_l\boldsymbol{e}_l^T\right)^{-1} \\ &= \bar{\boldsymbol{\Sigma}}_l - \frac{\bar{\boldsymbol{\Sigma}}_l\boldsymbol{e}_l\boldsymbol{e}_l^T\bar{\boldsymbol{\Sigma}}_l}{\hat{\alpha}_l^{-1} + \boldsymbol{e}_l^T\bar{\boldsymbol{\Sigma}}_l\boldsymbol{e}_l},\end{aligned} \tag{19}$$

where the latter expression was obtained using the matrix inversion lemma [5] and defining

$$\bar{\boldsymbol{\Sigma}}_l = \left(\hat{\tau}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \sum_{j\neq l}\hat{\alpha}_j\boldsymbol{e}_j\boldsymbol{e}_j^T\right)^{-1}. \tag{20}$$

Finally we define

$$\varsigma_l = \boldsymbol{e}_l^T\bar{\boldsymbol{\Sigma}}_l\boldsymbol{e}_l \quad \text{and} \quad \rho_l^2 = \boldsymbol{e}_l^T\bar{\boldsymbol{\Sigma}}_l\mathbf{B}\bar{\boldsymbol{\Sigma}}_l\boldsymbol{e}_l. \tag{21}$$

Now, by plugging (19) in (18) and (17) and using the definitions (21) we obtain

$$\hat{\alpha}_l^{-1} = \rho_l^2 + \varsigma_l - \frac{\varsigma_l^2 + 2\varsigma_l\rho_l^2}{\hat{\alpha}_l^{-1} + \varsigma_l} + \frac{\varsigma_l^2\rho_l^2}{(\hat{\alpha}_l^{-1} + \varsigma_l)^2}. \tag{22}$$

Expression (22) is a modified version of (17) that is now an implicit function of $\hat{\alpha}_l$. Solving for $\hat{\alpha}_l$ naturally leads to the desired fixed point $\hat{\alpha}_l^{[\infty]}$. Observe that (22) can be seen as a nonlinear map $\hat{\alpha}_l^{[m+1]} = F(\hat{\alpha}_l^{[m]})$ that at iteration m maps $\hat{\alpha}_l^{[m]}$ to $\hat{\alpha}_l^{[m+1]}$. Naturally, the fixed points of this map are equivalent to the desired (possibly multiple) fixed points $\hat{\alpha}_l^{[\infty]}$.

The following theorem provides analytical expressions for the fixed points of the map $\hat{\alpha}_l^{[m+1]} = F(\hat{\alpha}_l^{[m]})$.

Theorem 1. *Assuming an initial condition $\alpha_l^{[0]} \geq 0$, the iterations of the nonlinear map*

$$\begin{aligned}\hat{\alpha}_l^{[m+1]} &= F(\hat{\alpha}_l^{[m]}) \\ &= \left(\rho_l^2 + \varsigma_l - \frac{\varsigma_l^2 + 2\varsigma_l \rho_l^2}{\frac{1}{\hat{\alpha}_l^{[m]}} + \varsigma_l} + \frac{\varsigma_l^2 \rho_l^2}{\left(\frac{1}{\hat{\alpha}_l^{[m]}} + \varsigma_l\right)^2} \right)^{-1},\end{aligned} \tag{23}$$

where ρ_l^2 and ς_l are given by (21)*, converge as $m \to \infty$ to one of the following fixed points $\hat{\alpha}_l^{[\infty]}$:*

$$\hat{\alpha}_l^{[\infty]} = \begin{cases} (\rho^2 - \varsigma)^{-1} & \rho^2 > \varsigma \\ \infty & \rho^2 \leq \varsigma \end{cases} \tag{24}$$

Proof. We begin the proof by first computing the fixed points of the map $\hat{\alpha}_l^{[m+1]} = F(\hat{\alpha}_l^{[m]})$. By inspecting (22) we observe that $\hat{\alpha}_l^{[\infty]} = \infty$ is a fixed point of the map. The other solution is found by solving $\hat{\alpha}_l^* - F(\hat{\alpha}_l^*) = 0$ with respect to $\hat{\alpha}_l^*$. After rather tedious but straightforward algebraic manipulations we obtain the second fixed point at

$$\hat{\alpha}_l^{[\infty]} = \hat{\alpha}_l^* = (\rho_l^2 - \varsigma_l)^{-1} \tag{25}$$

We now investigate the stability of the fixed point (25) by analyzing the asymptotic stability of the map (23) in the vicinity of $\hat{\alpha}_l^{[\infty]} = (\rho_l^2 - \varsigma_l)^{-1}$. It is known that a fixed point of a map is asymptotically stable if the eigenvalues of the Jacobian of the map evaluated at this fixed point are all within the unit circle. The Jacobian of $F(\hat{\alpha}_l^{[m]})$ evaluated at (25) is given by

$$\left. \frac{\mathrm{d}F(\hat{\alpha}_l^{[m]})}{\mathrm{d}\hat{\alpha}_l^{[m]}} \right|_{\hat{\alpha}_l^{[m]} = (\rho_l^2 - \varsigma_l)^{-1}} = -\frac{\varsigma_l(\varsigma_l - 2\rho_l^2)}{\rho_l^4}. \tag{26}$$

Now it is straightforward to show that

$$\left| \frac{\varsigma_l(\varsigma_l - 2\rho_l^2)}{\rho_l^4} \right| < 1,$$

i.e., that (25) is a stable fixed point of the map, when ρ_l^2 and ς_l satisfy the following inequality constraints

$$\rho_l^2 > \varsigma_l, \tag{27}$$

$$\rho_l^2 < \varsigma_l < (1 + \sqrt{2})\rho_l^2. \tag{28}$$

Observe that the condition (28) suggests that the fixed point (25) might become negative. However, the negative value of $\alpha_l^{[m]}$ cannot be reached for $\alpha_l^{[0]} \geq 0$ since the map (23) can be shown to be positive for ς_l and ρ_l^2 satisfying (28).[7] Thus, $\hat{\alpha}_l^{[\infty]} = (\rho_l^2 - \varsigma_l)^{-1}$ is a stable positive fixed point only when $\rho_l^2 > \varsigma_l$; if $\varsigma_l > \rho_l^2$, then (25) looses its stability and the iterations of the map converge to the other positive fixed point at $\hat{\alpha}_l^{[\infty]} = \infty$, i.e., the iterations simply diverge. $\square$

Expression (25) together with the pruning condition (27) is a variational counterpart of the marginal likelihood analysis performed in [11]. It allows one to assess the impact of the lth basis vector $\boldsymbol{\varphi}_l$ in the matrix $\boldsymbol{\Phi}$ on the variational lower bound by computing (24): finite value of $\hat{\alpha}_l^{[\infty]}$ instructs us to keep the lth component since it should maximize the variational lower bound, while the infinite value of $\hat{\alpha}_l^{[\infty]}$ indicates that the basis vector l is superfluous. In this way all L basis vectors can be processed sequentially in a round-robin fashion. It is also evident that the decision to prune or to keep a basis function is determined entirely based on the target vector $\boldsymbol{t}$ and the weight covariance matrix $\hat{\boldsymbol{\Sigma}}$.

3.2 Efficient Implementation of Fast Variational SBL

Let us assume that at some iteration j of the algorithm we have L basis functions and an estimate of $\hat{\boldsymbol{\Sigma}}$. Our goal is to test whether the lth basis function $\psi_l(\cdot)$ should be kept in the model.

The matrix $\hat{\boldsymbol{\Sigma}}$ can be efficiently computed using rank one updates. Indeed, noting that $\hat{\boldsymbol{\Sigma}}^{-1} = \hat{\tau}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \sum_{l=1}^{L} \hat{\alpha}_k \boldsymbol{e}_k \boldsymbol{e}_k^T$ we compute $\bar{\boldsymbol{\Sigma}}_l$ as (see [6])

$$\bar{\boldsymbol{\Sigma}}_l = \left(\hat{\boldsymbol{\Sigma}}^{-1} - \hat{\alpha}_l \boldsymbol{e}_l \boldsymbol{e}_l^T\right)^{-1} = \hat{\boldsymbol{\Sigma}} + \frac{\hat{\boldsymbol{\Sigma}}\boldsymbol{e}_l\boldsymbol{e}_l^T\hat{\boldsymbol{\Sigma}}}{\hat{\alpha}_l^{-1} - \boldsymbol{e}_l^T\hat{\boldsymbol{\Sigma}}\boldsymbol{e}_l}. \tag{29}$$

Then, if the test indicates that the basis function $\psi(\cdot)$ should be removed form the model, we can immediately update $\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{\Sigma}}_{\bar{l}}$ with

$$\hat{\boldsymbol{\Sigma}}_{\bar{l}} = \left[\hat{\boldsymbol{\Sigma}} - \frac{\hat{\boldsymbol{\Sigma}}\boldsymbol{e}_l\boldsymbol{e}_l^T\hat{\boldsymbol{\Sigma}}}{\boldsymbol{e}_l^T\hat{\boldsymbol{\Sigma}}\boldsymbol{e}_l}\right]_{\bar{l},\bar{l}}, \tag{30}$$

which is equivalent to computing $\hat{\boldsymbol{\Sigma}}$ using (13) with the basis $\boldsymbol{\varphi_l}$ removed from the model. We have used the notation $[\cdot]_{\bar{l},\bar{l}}$ in (30) to denote the matrix obtained by removing the lth row and column. Alternatively, if the test indicates that the basis $\boldsymbol{\varphi}_l$ should be retained in the model, the covariance matrix $\hat{\boldsymbol{\Sigma}}$ is updated using (19).

We now summarize the main steps of the proposed fast variational SBL method in Algorithm 1 and introduce the vector notation $[\cdot]_{\bar{l}}$ to denote a vector obtained by removing the lth element.

[7]Naturally, the map (23) is also positive for ς_l and ρ_l^2 satisfying (27)

Algorithm 1

```
Initialize $q(\boldsymbol{w})$, $q(\boldsymbol{\alpha})$, $q(\tau)$
while Continue if not converged do
    for $l \in \{1, \dots, L\}$ do
        Compute: $\bar{\boldsymbol{\Sigma}}_l$ from (29) and $\varsigma_l$ and $\rho_l^2$ from (21)
        if $\rho_l^2 > \varsigma_l$ then
            $\hat{\alpha}_l = 1/(\rho_l^2 - \varsigma_l)$
            Update $\hat{\boldsymbol{\Sigma}}$ from (19)
        else
            Update $\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{\Sigma}}_{\bar{l}}$ from (30), $\hat{\boldsymbol{\alpha}} = [\hat{\boldsymbol{\alpha}}]_{\bar{l}}$, $L = L - 1$
        end if
    end for
    Compute $\hat{\boldsymbol{\mu}}$ from (14) and update $q(\tau)$ from (16)
    Check for convergence
end while
```

3.3 Simulations

In this section, we compare the simulation results of our proposed algorithm with other existing SBL methods, namely the standard RVM [10], the variational RVM [3] and the fast marginal likelihood maximization method [11]. Two experiments are presented. Section 3.3.1 shows a common *sinc* regression problem and in Section 3.3.2 and 3.3.3 we test the algorithms on a random basis with only a few active components. Note that for better comparison of the algorithms, we have also implemented the fast methods in a "just pruning" manner which is different from the proposed method in [11], where basis functions can also be added to the model. Thus, all methods start with the full design matrix $\boldsymbol{\Phi}$ and delete columns according to their individual sparsification criteria. For the methods [3] and [10] one needs to specify a threshold on the hyperparmeters $\hat{\alpha}_l$, $l = 1, \dots, L$, to numerically detect if they have diverged, where for this case when one $\hat{\alpha}_l$ exceeds the threshold, the appropriate basis function is pruned from the model. Usually this threshold it set to a very large number, e.g. 10^{12}, but for slower methods like the variational RVM [3] we have observed that this would need an prohibitive long amount of simulation time and thus have restricted the threshold to 10^4 for this particular method although fully aware of the influence on the simulation results. To show the effect of the threshold level onto the results, we have used two versions of the faster standard RVM [10] with two different thresholds. To circumvent the influence of the individual noise estimation procedures on the output, we assume that we know the true underlying noise variance $\hat{\tau}^{-1} = \sigma^2$ used for the simulations. The hyperparameter-mean $\hat{\boldsymbol{\alpha}}$ is initialized to $\breve{\alpha}_l = 1/(\breve{\mu}_l^2 + \breve{s}_l)$, $\forall l$, where the weights $\breve{\mu}_l$ are computed using a matching pursuit algorithm and the variance is given by $\breve{s}_l = \sigma^2/(\boldsymbol{\varphi}_l^T \boldsymbol{\varphi}_l)$. Furthermore, since the fast methods need to have a sequence in which they update the elements of the vector $\hat{\boldsymbol{\alpha}}$, the inverse ordering of the matching pursuit ranking is used, i.e. the worst aligned basis will be updated first. Note that for the standard RVM [10] and the variational RVM [3] no such ordering is needed. We have used the same convergence criteria for all methods, where the algorithm stops if the ℓ^2-norm of the hyperparameter-difference between two consecutive iteration steps $||\hat{\boldsymbol{\alpha}}_{\text{new}} - \hat{\boldsymbol{\alpha}}_{\text{old}}||$ is smaller than 10^{-3}.

3.3.1 Sinc regression

The function $\text{sinc}(x) = \sin(x)/x$ is a very common nonlinear evaluation function used for regression. We also would like to focus our attention on this function for one-dimensional inputs x. To generate the training data we drew $N = 100$ samples x_n, $n = 1, \dots, N$, from a uniform distribution over the interval $[-10, 10]$. The targets t_n are then generated by adding white gaussian noise to the function, i.e. $t_n = \text{sinc}(x_n) + \epsilon_n$, where $\epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2)$. We used a constant bias $b(x) = 1$ and N Gaussian kernels $\kappa(x, x_n) = \exp\{-(x - x_n)^2/2\nu^2\}$ centered on each sample x_n as the basis functions. The kernel variance ν^2 was set to 2.3 for all algorithms. The initial size of the design matrix $\mathbf{\Phi}$ thus is $N \times (N + 1)$, where the n-th row of $\mathbf{\Phi}$ is defined as $[1, \kappa(x_n, x_1), \kappa(x_n, x_2), \dots, \kappa(x_n, x_N)]$.

From the plots (a) and (b) of Figure 3 we see that our proposed algorithm achieves comparable results to the existing SBL methods in terms of model error and sparsity but has a convergence speed at the level of the fast marginal likelihood maximization method [11] as depicted in (c).

3.3.2 Random basis

In this experiment we generate $L = 100$ basis vectors randomly for $N = 100$ samples drawn from a normal distribution $\boldsymbol{\varphi}_l \sim \mathcal{N}(\boldsymbol{\varphi}_l|\mathbf{0}, \mathbf{I}_N)$, $l = 1, \dots, L$. The targets are generated using a noisy version of the superposition of five randomly picked basis vectors, where $\boldsymbol{t} = \sum_{j \in \mathcal{R}_5} \boldsymbol{\varphi}_j + \boldsymbol{\xi}$, $\mathcal{R}_5$ is a set of five randomly picked basis indices and $\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \sigma^2\mathbf{I}_N)$ is a vector containing the additive white Gaussian noise samples.

The simulation results are presented in Figure 4. We observe that methods using a lower threshold perform better for higher SNR values than those with a higher threshold. Considering a threshold of 10^4 it is interesting to mention that the standard RVM [10] and the variational RVM [3] actually reach the true number of basis vectors above a certain level of SNR. Note that the fast methods indirectly incorporate a threshold that is infinite with their specific pruning criteria.

In the following section we show how our method can be improved by slightly modifying the pruning criteria, i.e. not fullfilling the inequality (27).

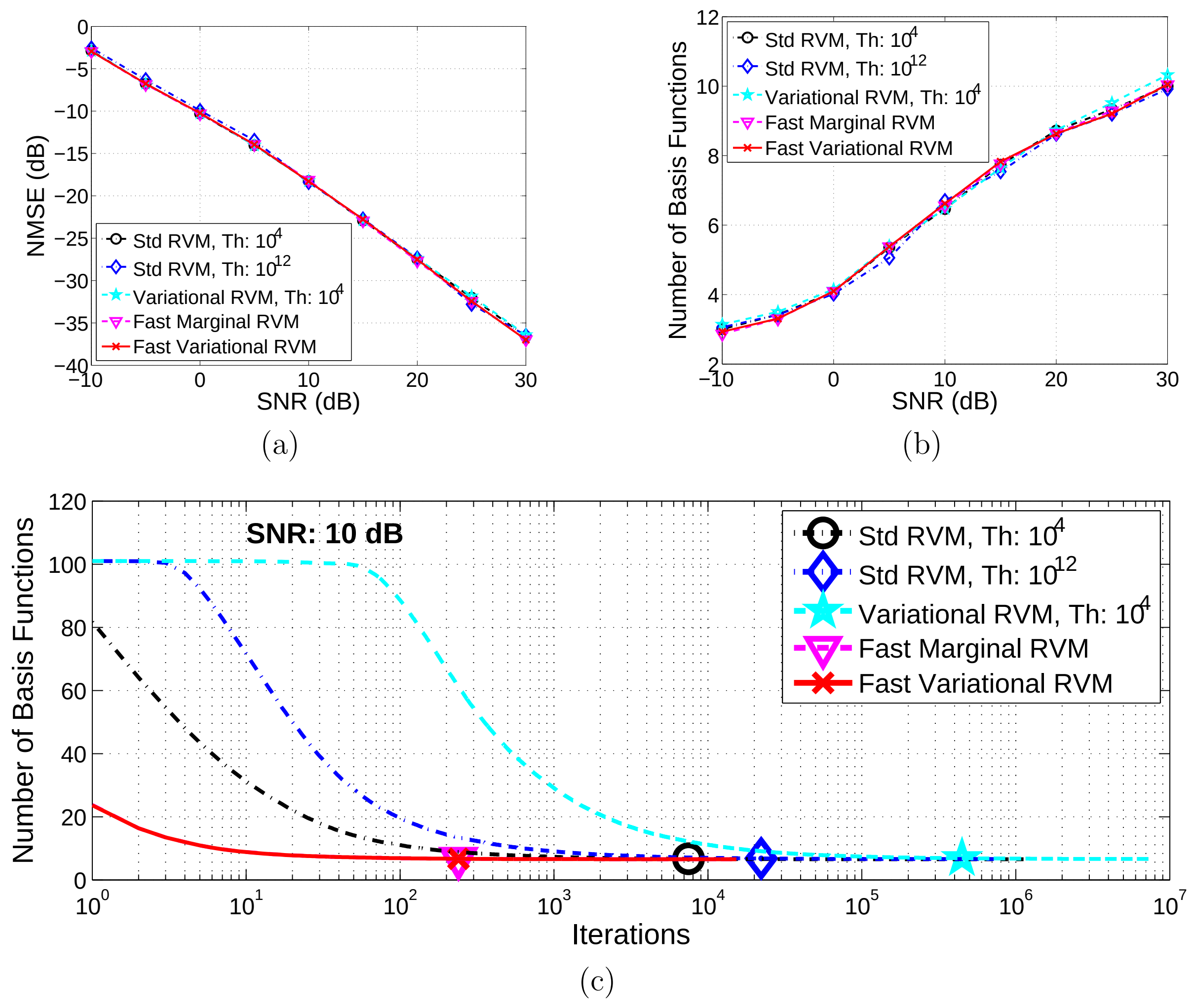
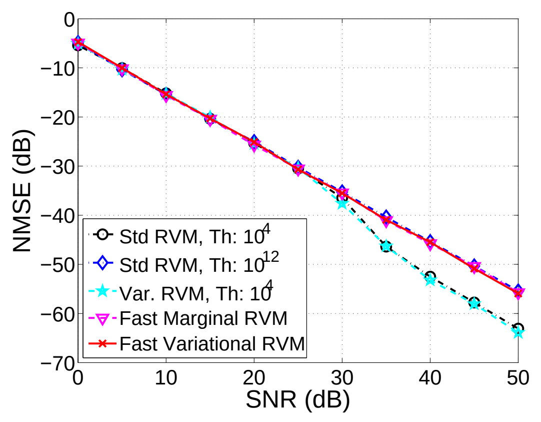


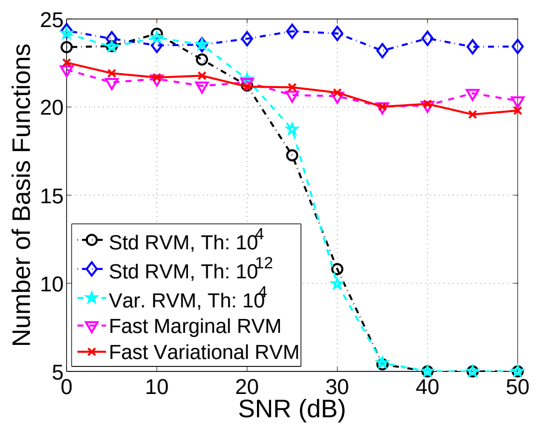
Figure 3: Sinc regression performance for different sparse Bayesian learning (SBL) methods. All results are averaged over 500 realizations. Plot (a) gives the normalized mean squared error (NMSE) over the signal to noise ratio (SNR) and plot (b) the number of basis functions over the SNR. Plot (c) shows the progress of the number of basis functions over iteration-steps for an SNR of 10dB. The markers depict the average number of iterations.

3.3.3 Random basis with adjusted pruning criteria

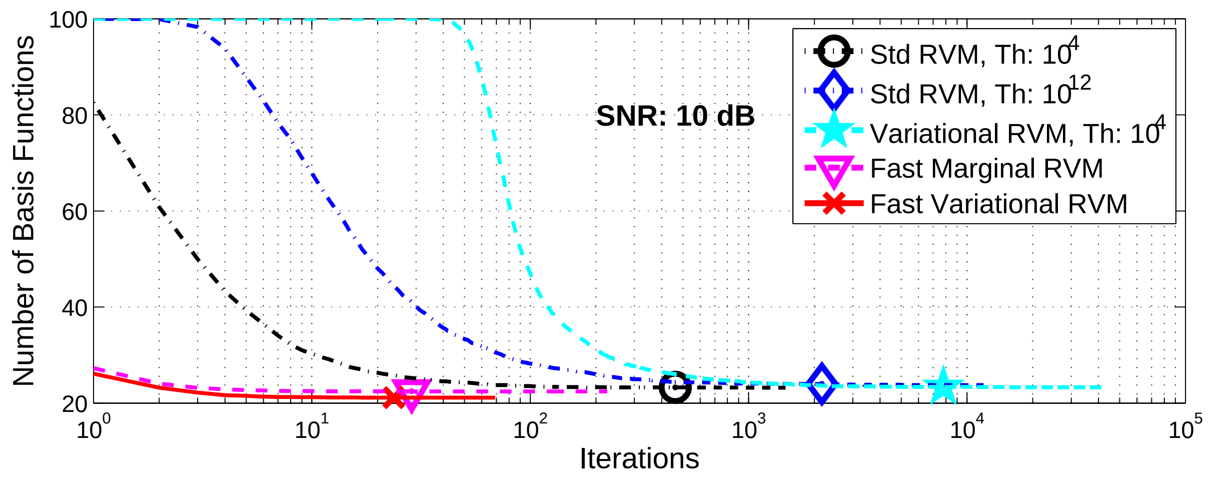
Since the pruning condition (27) is a key to the model sparsity, let us study it in more details. Assume for the moment that $\mathbf{\Phi} = [\boldsymbol{\varphi}_1]$, i.e., that we only have a single basis function $\psi_1(\cdot)$. In this case $\hat{\Sigma} = (\hat{\tau}\boldsymbol{\varphi}_1^T\boldsymbol{\varphi}_1 + \hat{\alpha}_1)^{-1}$ and $\bar{\Sigma}_1 = (\hat{\tau}\boldsymbol{\varphi}_1^T\boldsymbol{\varphi}_1)^{-1}$. It then follows from (21) that $\rho_1^2 = |\hat{\tau}\bar{\Sigma}_1\boldsymbol{\varphi}_1^T\boldsymbol{t}|^2$ and $\varsigma_1 = \bar{\Sigma}_1$. These quantities correspond respectively to a squared weight mean $\bar{\mu}_1^2 \equiv \rho_1^2$ of the basis $\boldsymbol{\varphi}_1$ and the estimated variance of this weight $\bar{s}_1 \equiv \varsigma_1$ obtained when $\hat{\alpha}_1 \equiv 0$. Obviously, evaluating (27) is equivalent to comparing the squared weight mean $\bar{\mu}_1^2$ to its estimated variance $\bar{s}_1$; also, the ratio $\bar{\mu}_1^2/\bar{s}_1 = \bar{\mu}_1^2\hat{\tau}\boldsymbol{\varphi}_1^T\boldsymbol{\varphi}_1$ can be recognized as an estimate of the signal-to-noise ratio (SNR) for the basis vector $\boldsymbol{\varphi}_1$. Furthermore, according to (27) the basis $\boldsymbol{\varphi}_1$ is retained in the model provided $SNR_1 = \omega_1^2/\varsigma_1 > 1$, i.e., when the component's SNR



(a)



(b)



(c)

Figure 4: Random basis performance, where only 5 of 100 basis vectors are active. All results are averaged over 50 realizations. Plot (a) gives the normalized mean squared error (NMSE) over the signal to noise ratio (SNR) and plot (b) the number of basis functions over the SNR. Plot (c) shows the progress of the no. of basis functions over iteration-steps for an SNR of 10dB. The markers depict the average number of iterations.

is above 0dB.

This interpretation of the pruning condition (27) remains valid also for a general design matrix $\mathbf{\Phi}$ with more than one basis vector. For a particular basis $\boldsymbol{\varphi}_l$ the parameters ρ_l^2 and ς_l can be seen respectively as the estimated squared weight of $\boldsymbol{\varphi}_l$ and the weight's variance computed when $\hat{\alpha}_l = 0$ and $\hat{\alpha}_k$, $k \neq l$, are fixed. As a consequence, $SNR_l = \rho_l^2/\varsigma_l$ defines the SNR of the basis $\boldsymbol{\varphi}_l$ when the corresponding sparsity parameter $\hat{\alpha}_l = 0$.

This simple interpretation of the pruning test can be used to generalize (27) to any desired $SNR_l > 1$. More specifically, given a certain desired $SNR' \geq 1$, the pruning (27) can be empirically adjusted as

$$\rho_l^2 > \varsigma_l \times SNR'. \tag{31}$$

The modified condition (31) allows to remove all components with SNR_l satisfying $1 < SNR_l \leq SNR'$. Note, however, that this adjustment might potentially "harm" the variational lower bound (11), since it will remove basis functions with finite sparsity parameters. Nonetheless, this strategy might be of practical interest in scenarios where the true SNR is known and the goal is to delete spurious components introduced by SBL due to the "imperfectness" of the Gaussian sparsity prior.

Figure 5 compares the adjusted sparsification criteria with the other methods, where we have set SNR$'$ of condition (31) to the true SNR. For comparison, we also have plotted the original condition which corresponds to a setting of SNR$' = 0$dB. As could be seen in plot (a) and (b), the modified version achieves a lower NMSE and correctly detects the real number of basis functions for SNR values larger than about 10dB. The outcome is a tremendous speedup since the algorithm converges to the optimum in not more than three iterations on average as could be seen from (c) for the case of SNR = 10dB.

4 Publications and Ongoing Research

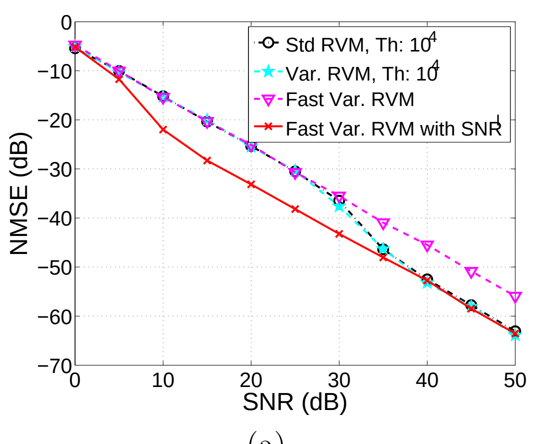
We have already submitted 3 papers that were built on the work described in Section 3:

- D. Shutin, T. Buchgraber, S. R. Kulkarni, and H. V. Poor, "Fast variational sparse Bayesian learning with automatic relevance determination," submitted to IEEE Transactions on Signal Processing, 2010
- D. Shutin, T. Buchgraber, S. R. Kulkarni and H. V. Poor, "Fast adaptive variational sparse Bayesian learning with automatic relevance determination," submitted to ICASSP'11, 2010
- T. Buchgraber, D. Shutin and H. Vincent Poor, "A sliding-window online fast variational sparse Bayesian learning algorithm," submitted to ICASSP'11, 2010,

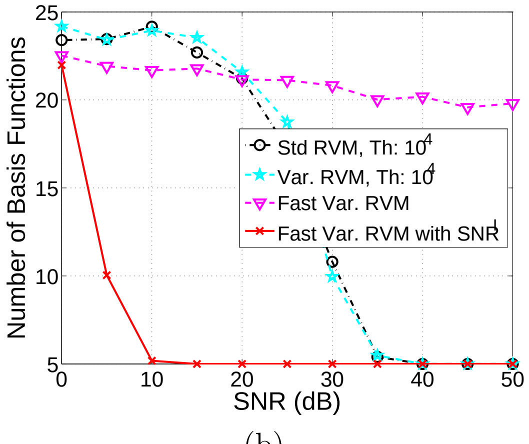
which is one journal and two conference papers.

In the journal paper "Fast variational sparse Bayesian learning with automatic relevance determination" we generally descibe the results obtained during the collaboration at Princeton University mentioned in Section 3.

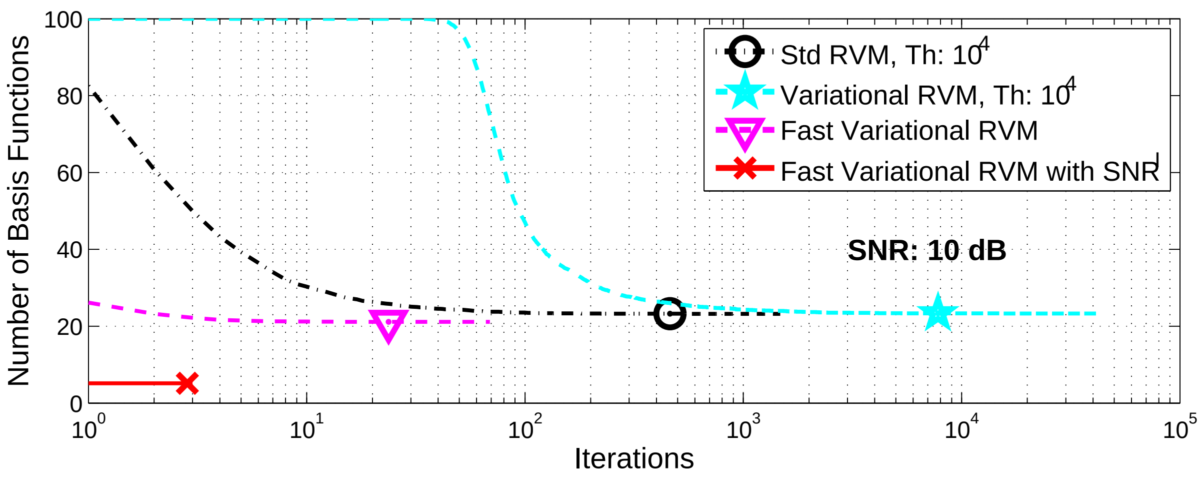
The publication about "Fast adaptive variational sparse Bayesian learning with automatic relevance determination" is about a dynamically adding and pruning concept for basis functions which is related to the work in [11], a similar method that maximizes a marginal-likelihood function instead of a variational lower bound. The case of testing the basis functions currently included in the model has already been discussed in Section 3.2 and used in Algorithm 1. We can rewrite the elementary decision rule for keeping or pruning the lth basis function as given in



(a)



(b)



(c)

Figure 5: Random basis performance, where only 5 of 100 basis vectors are active. Comparison of SBL methods with the proposed adjusted pruning criterion according to condition (31) using $SNR' = SNR$, the true signal to noise ratio. All results are averaged over 50 realizations. Plot (a) gives the normalized mean squared error (NMSE) over the SNR and plot (b) the number of basis functions over the SNR. Plot (c) shows the progress of the number of basis functions over the iteration-steps for an SNR of 10dB. The markers depict the average number of iterations.

Algorithm 2. We use the notation $\hat{\alpha}_l^{\text{old}}$ to denote the previous value of $\hat{\alpha}_l$. When a basis function is kept, we use the matrix inversion lemma [5] to efficiently add the term $(\hat{\alpha}_l - \hat{\alpha}_l^{\text{old}})\boldsymbol{e}_l\boldsymbol{e}_l^T$ to the inverse of the weight covariance matrix $\hat{\boldsymbol{\Sigma}}$, which is a rank-one update of the lth hyperparameter. To test a new candidate basis function for an existing model containing L basis vectors in the columns of the design matrix $\boldsymbol{\Phi}$ we compute

$$\bar{\boldsymbol{\Sigma}}_{L+1} = \begin{pmatrix} \hat{\tau}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \operatorname{diag}(\hat{\boldsymbol{\alpha}}) & \hat{\tau}\boldsymbol{\Phi}^T\boldsymbol{\varphi}_{L+1} \\ \hat{\tau}\boldsymbol{\varphi}_{L+1}^T\boldsymbol{\Phi} & \hat{\tau}\boldsymbol{\varphi}_{L+1}^T\boldsymbol{\varphi}_{L+1} \end{pmatrix}^{-1}, \tag{32}$$

Algorithm 2 Testing the lth basis $\boldsymbol{\varphi}_l$

```
Compute $\varsigma_l$ and $\rho_l^2$ from (21)
if $\rho_l^2 > \varsigma_l$ then
    % keep basis
    $\hat{\alpha}_l = (\rho_l^2 - \varsigma_l)^{-1}$,  $\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{\Sigma}} - \frac{\hat{\boldsymbol{\Sigma}}\boldsymbol{e}_l\boldsymbol{e}_l^T\hat{\boldsymbol{\Sigma}}}{(\hat{\alpha}_l - \hat{\alpha}_l^{\text{old}})^{-1} + \boldsymbol{e}_l^T\hat{\boldsymbol{\Sigma}}\boldsymbol{e}_l}$
else
    % prune basis
    Compute $\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{\Sigma}}_{\bar{l}}$ from (30)
    $\hat{\boldsymbol{\alpha}} = [\hat{\boldsymbol{\alpha}}]_{\bar{l}}$,  $\boldsymbol{\Phi} = [\boldsymbol{\Phi}]_{:,\bar{l}}$,  $L = L - 1$
end if
```

which is equivalent to (20) if $\boldsymbol{\Phi}$ would have been extended by the basis vector $\boldsymbol{\varphi}_{L+1}$ in an additional column at the end. By using the inversion rule for structured matrices [9], (32) is equivalent to

$$\bar{\boldsymbol{\Sigma}}_{L+1} = \begin{pmatrix} \mathbf{V} & -\gamma\hat{\tau}\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{\varphi}_{L+1} \\ -\gamma\hat{\tau}\boldsymbol{\varphi}_{L+1}^T\boldsymbol{\Phi}\hat{\boldsymbol{\Sigma}} & \gamma \end{pmatrix} \tag{33}$$

where $\mathbf{V} = \hat{\boldsymbol{\Sigma}} + \gamma\hat{\tau}^2\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{\varphi}_{L+1}\boldsymbol{\varphi}_{L+1}^T\boldsymbol{\Phi}\hat{\boldsymbol{\Sigma}}$ and

$$\gamma = (\hat{\tau}\boldsymbol{\varphi}_{L+1}^T\boldsymbol{\varphi}_{L+1} - \hat{\tau}^2\boldsymbol{\varphi}_{L+1}^T\boldsymbol{\Phi}\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{\varphi}_{L+1})^{-1}. \tag{34}$$

Similarly we can efficiently compute $\hat{\boldsymbol{\Sigma}}_{L+1}$ by using $\lambda = (\gamma^{-1} + \hat{\alpha}_{L+1})^{-1}$, and $\mathbf{W} = \hat{\boldsymbol{\Sigma}} + \lambda\hat{\tau}^2\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{\varphi}_{L+1}\boldsymbol{\varphi}_{L+1}^T\boldsymbol{\Phi}\hat{\boldsymbol{\Sigma}}$ or by updating $\bar{\boldsymbol{\Sigma}}_{L+1}$ with

$$\begin{aligned} \hat{\boldsymbol{\Sigma}}_{L+1} &= \begin{pmatrix} \mathbf{W} & -\lambda\hat{\tau}\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{\varphi}_{L+1} \\ -\lambda\hat{\tau}\boldsymbol{\varphi}_{L+1}^T\boldsymbol{\Phi}\hat{\boldsymbol{\Sigma}} & \lambda \end{pmatrix} \\ &= \bar{\boldsymbol{\Sigma}}_{L+1} - \frac{\bar{\boldsymbol{\Sigma}}_{L+1}\boldsymbol{e}_{L+1}\boldsymbol{e}_{L+1}^T\bar{\boldsymbol{\Sigma}}_{L+1}}{\hat{\alpha}_{L+1}^{-1} + \boldsymbol{e}_{L+1}^T\bar{\boldsymbol{\Sigma}}_{L+1}\boldsymbol{e}_{L+1}} \end{aligned}. \tag{35}$$

The Algorithm 3 summarizes the main steps for adding or rejecting a new candidate basis function. By starting with one arbitrary basis function $\psi^*(\cdot)$, the design matrix is defined as

Algorithm 3 Testing a new basis $\boldsymbol{\varphi}_{L+1}$

```
Compute $\varsigma_{L+1}$ and $\rho_{L+1}^2$ from (21) using $\bar{\boldsymbol{\Sigma}}_{L+1}$ from (33)
if $\rho_{L+1}^2 > \varsigma_{L+1}$ then
    % add new basis
    $\hat{\alpha}_{L+1} = (\rho_{L+1}^2 - \varsigma_{L+1})^{-1}$
    Compute $\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{\Sigma}}_{L+1}$ using (35)
    $\boldsymbol{\Phi} = [\boldsymbol{\Phi}, \boldsymbol{\varphi}_{L+1}]$, $\hat{\boldsymbol{\alpha}} = [\hat{\boldsymbol{\alpha}}^T, \hat{\alpha}_{L+1}]^T$, $L = L + 1$
else
    % reject new basis - no action needed
end if
```

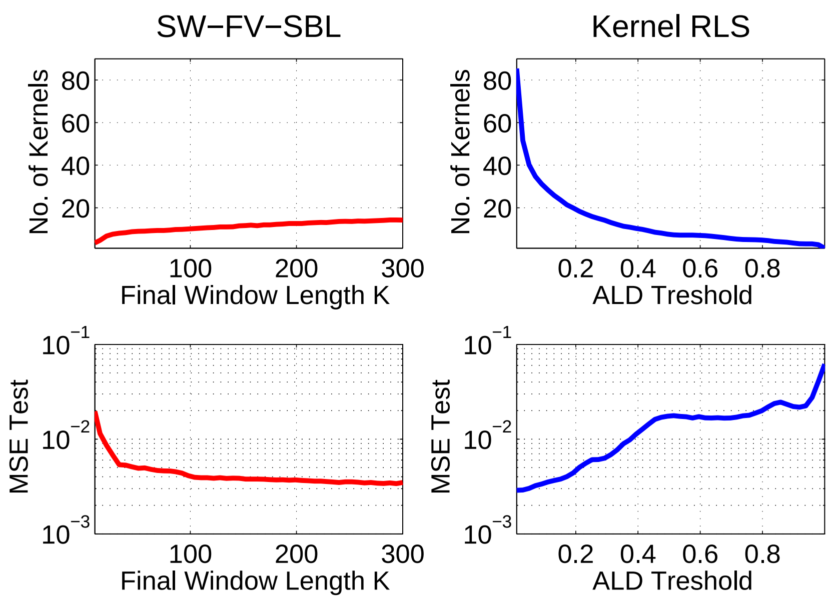


Figure 6: Comparison of the sliding window fast variational SBL algorithm with an ALD Kernel-RLS for one-step ahead prediction of Mackey-Glass data. Results are averaged over 200 independent realizations for 500 samples. For the MSE, a test set of 200 samples was used.

$\mathbf{\Phi} = [\boldsymbol{\varphi}^*]$. We can add basis functions accoring to Algorithm 3 and test the functions in the model with Algorithm 2. This gives a very efficient method because the inversion in (13) only has to be performed on a small number of basis functions at each iteration. This is opposed to the method presented in Section 3, where we start with all basisfunctions and then sparsify the model according to the pruning condition.

An online version of the proposed sparsification method is presented in the paper "A sliding-window online fast variational sparse Bayesian learning algorithm", where we use a sliding-window to evaluate the design matrix $\mathbf{\Phi}_n$ and the target vector $\boldsymbol{t}_n$, which now depend on a time index n. This method also uses the previous mentioned concept of dynamically adding and pruning basis functions. Figures 6 and 7 show the performance of the online method compared to a state of the art kernel recursive least squares (Kernel RLS) algorithm with approximate linear dependency (ALD) as a sparsification criteria [4]. To make both methods comparable we have also used Gaussian kernels as basis functions for our method. The algorithms are tested on the one-step ahead prediction ability of Mackey Glass chaotic time series data produced according to [7, Section 2.11.1]. The free design parameters of both methods are the sliding window length K for the fast variational method and a threshold parameter for the Kernel RLS. Figure 6 shows the influence on the setting of both parameters, where we can see that the Kernel RLS is more sensitive to changes. We can also see that for same values of test mean square error (MSE), our method achieves much sparser representations.

Based on the work at Princeton University we plan to lead the research into the following directions, i.e. to get closer to the main aim of distributed SBL. First we would like to focus on

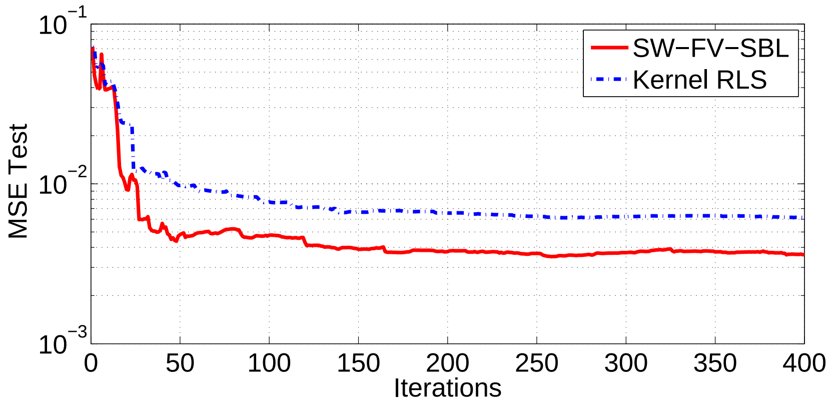


Figure 7: Example learning curves of the sliding window fast variational SBL algorithm using a window length of 300 samples and the Kernel-RLS using an ALD threshold of 0.3. Both methods result in the same number of 14 kernels at the last iteration. For the MSE, a test set of 200 samples was used.

Online fast variational SBL, where we try to modify the sliding window online version of the algorithm. We have shown that by using Gaussian kernels we achieve sparser representations than a Kernel-RLS using ALD for equivalent SNR levels or on the other hand achieve a lower SNR for the same number of used kernels. We would like to replace the sliding-window by evaluations directly on the kernel centers, thus constucting a quatratic design matrix similar to a Gram matrix. This should lead to a method containing no free design parameters, like in our case the sliding-window length. Thus, the outcoming method should get more flexible for different learning tasks and no compromisses concerning the sliding window length are needed. The second topic of future research is **Distributed sparse Bayesian learning**. As mentioned in Section 1 the general aim of all the described methods is to exploit SBL for distributed data processing in WSNs. Here we would like to investigate if our efficient sparsification criteria can be incorporated in distributed sensor environments. We seek a way to implement our criteria in distributed algorithms like consensus propagation [8] or average consensus [12]. Such consensus schemes have no need for routing and the nodes just have to know their direct neighbours with which they have to communicate. It can also be thought of merging relevant basis functions using only local communications in a sensor network. Also the previously mentioned online kernel version should be tested on a sequential spatial sensor chain instead of a time sequence.

References

[1] Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. A survey on sensor networks. *Communications Magazine, IEEE 40*, 8 (Aug 2002), 102 – 114.

[2] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and*

Statistics), 1st ed. 2006. corr. 2nd printing ed. Springer, October 2007.

[3] Bishop, C. M., and Tipping, M. E. Variational relevance vector machines. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence* (San Francisco, CA, USA, 2000), Morgan Kaufmann Publishers Inc., pp. 46–53.

[4] Engel, Y., Mannor, S., and Meir, R. The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing 52*, 8 (Aug. 2004), 2275 – 2285.

[5] Golub, G. H., and Van Loan, C. F. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences - 3rd Edition)*, 3rd ed. The Johns Hopkins University Press, October 1996.

[6] Hager, W. W. Updating the inverse of a matrix. *SIAM Review 31*, 2 (1989), pp. 221–239.

[7] Liu, W., Principe, J. C., and Haykin, S. *Kernel Adaptive Filtering: A Comprehensive Introduction.* Wiley Publishing, 2010.

[8] Moallemi, C. C., and Roy, B. V. Consensus propagation. *IEEE Transactions on Information Theory 52* (2006), 4753–4766.

[9] Petersen, K. B., and Pedersen, M. S. The matrix cookbook, October 2008. Version 20081110.

[10] Tipping, M. E. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research 1* (June 2001), 211–244.

[11] Tipping, M. E., and Faul, A. C. Fast marginal likelihood maximisation for sparse bayesian models. In *roceedings of the Ninth International Workshop on Artificial Intelligence and Statistics* (Key West, FL,, January 2003).

[12] Xiao, L. A scheme for robust distributed sensor fusion based on average consensus. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN* (2005), pp. 63–70.